Hewlett-Packard -- Portable Computer Division

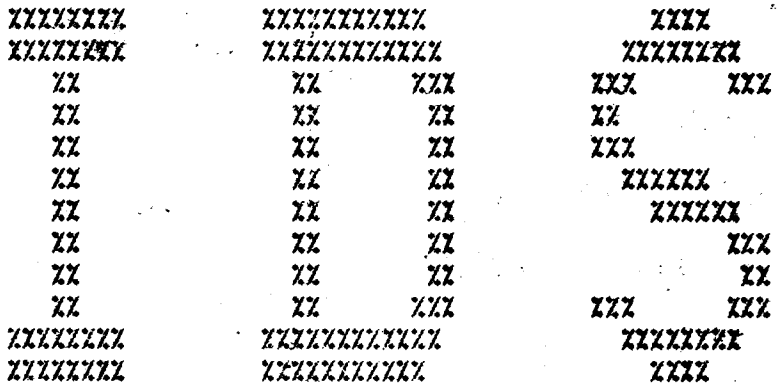
Corvallis, Oregon

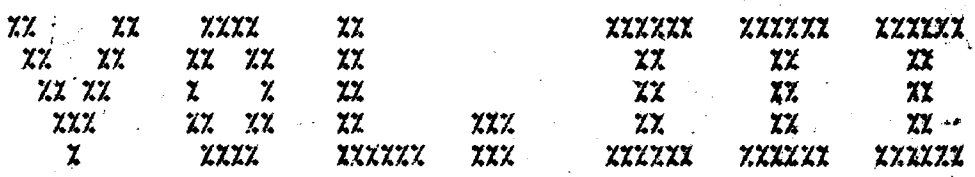
HP-71 Software

Internal Design Specification

VOLUME 3

Operating System Source Listings





HP Part No. 00071-90070

ROM Release 1B88B -- December 1983

(c) Copyright Hewlett-Packard Company 1983

**** NOTICE ****

Hewlett-Packard Company makes no express or implied warranty with regard to the documentation and program material offered or to the fitness of such material for any particular purpose. The documentation and program material is made available solely on an "as is" basis, and the entire risk as to its quality and performance is with the user. Should the documentation and program material prove defective, the user (and not Hewlett-Packard Company or any other party) shall bear the entire cost of all necessary correction and all incidental or consequential damages. Hewlett-Packard Company shall not be liable for any incidental or consequential damages in connection with or arising out of the furnishing, use, or performance of the documentation and program material.

Table of Contents

1 INTRODUCTION

2 LIST OF MODULES IN ADDRESS ORDER

3 LIST OF MODULES SORTED BY MODULE NAME

4 LOAD MAP FOR ROM RELEASE 1BBBB
Includes:
Module Summary
Cross Reference
Hex Dump Of Code

5 SOURCE MODULES IN ADDRESS ORDER

INTRODUCTION — CHAPTER 1

This volume contains the complete source code listings for the HP-71 Operating System. The modules are presented in address order according to their position in the operating system, from lowest address to highest address. The following sections give a list of the module names in address order, followed by an alphabetical list of the module names. A module's source file is denoted with an ampersand (&) in the file name, and its relocatable (binary) file with a percent sign (%) in the file name.

Interface information to an entry point or poll is described in a documentation header in the source file that contains that entry point or poll. Supported interfaces are identified by the "Name(S)" line of the documentation header, as described in Volume II of this document.

It is the intent of HP to preserve the supported interfaces described in this document, as well as the absolute address position of each supported entry point, through any future updates of the HP-71 operating system. In general this allows external software which uses these interfaces to work predictably without regard to the version of the HP-71 on which it is run. However, HP reserves the right to adjust the supported interfaces in any manner it chooses.

An unsupported entry point may be added to the supported list if HP deems the request to be justified. To request support for an entry point, please contact Systems Engineering Support in the HP Portable Computer Division Product Support Group at (503) 757-2000. Corrections or requested enhancements to the interface documentation are welcome and should also be reported in this manner.

WARNING !!

Only supported entry points are available for use by external software. HP expresses no intent to indefinitely preserve the interfaces to any unsupported entry points described in this volume or in Volume II, since it is inevitable that code in any 64K byte operating system will have to change or move occasionally to fix bugs. The interface to unsupported entry points, and their absolute address position, may therefore change at any time and without notice to outside parties.

LIST OF MODULES IN ADDRESS ORDER — CHAPTER 2

Address Range	Module	Title
00000 - 00784	SB%DVR	Main Driver
00785 - 00786	TI%REV	HP-71 Revision Number
00787 - 00C9A	SB%BIT	ASCII Tables (Rounded Letters)
00C9B - 00CF6	SB%FGT	F & G Shift Table
00CF7 - 01130	SB%KEY	Keyboard Driver (14 Column Keyboard)
01131 - 01626	TI%UTL	Command Execution Utilities
01627 - 0188F	SB%CMD	Command Stack Utilities
01890 - 02483	SB%DSP	Display Driver
02484 - 0259F	MB%ROM	HP-71 ROM Test
025A0 - 02C25	JP%PR1	Line Parse Driver - Part 1
02C26 - 033D5	JP%PR2	Parse Routines - Part 2
033D6 - 03FD4	JP%PR3	Parse Routines - Part 3
03FD5 - 048E5	SB%EXP	Expression Parser
048E6 - 04F5D	AB%LEX	Lexical Analyzer
04F5E - 05921	SG%LDC	Line Decompile
05922 - 0624F	SB%EXD	Expression Decompile
06250 - 062B1	TI%FX1	ROM 1 Fix Module
063A0 - 06F2E	SG%SYS	System Commands: DELETE, AUTO, and CAT
06F2F - 07DDB	JP%SYS	System Commands, RUN Loop
07DDC - 08647	JP%EXC	Program Execution Routines
08648 - 09392	SG%EXC	Miscellaneous Execution Routines
09393 - 09B75	TI%ERD	Error Message Driver
09B76 - 0A92A	SG%FXQ	File Execution Routines
0A92B - 0AA80	SB%FCN	Function Definitions
0AA81 - 0AA82	JP%ZER	Zero File
0AA83 - 0AA84	TI%REV	HP-71 Revision Number
0AA85 - 0AACE	TI%FX2	ROM 2 Fix Module
0ABF8 - 0B638	AB%REG	Variable Creation Routines
0B639 - 0C326	AB%FCN	Functions
0C327 - 0D434	JT%MTH	Math Routines - Part 1
0D435 - 0DE2E	SM%MTH	Math Routines - Part 2
0DE2F - 0E9F0	PM%STA	Statistics
0E9F1 - 0EC6D	MN%BP	BEEP Module
0EC6E - 0F177	MN%UTL	Miscellaneous Utilities
0F178 - 0F5D9	AB%EXP	Expression Execution Controller
0F5DA - 0FA46	AB%ASN	Assignment Statement Controller
0FA47 - 0FF5E	SC%TRC	TRACE Module
0FF5F - 10191	JP%MEM	Program Edit
10192 - 11052	MN%CNF	Configuration Module
11053 - 1231A	SC%FIL	File Utility & System Buffer Routines

Address Range	Module	Title
1231B - 12549	JP%POL	POLL Routine
1254A - 13448	MN%TM	Clock System
13449 - 138B4	PM%FLG	Flags, Traps, and Modes
138B5 - 14C7F	SC%DAT	Store & Retrieve Data - PRINT#/READ#
14C80 - 153A8	MN%ED	Character Editor
153A9 - 153AA	JP%ZER	Zero File
153AB - 153AC	TI%REV	HP-71 Revision Number
153AC	TI%FX3	ROM 3 Fix Module
1551F - 1632B	AB%CLC	CALC Mode
1632C - 16671	AB%BLD	CALC Mode Decompiler
16672 - 16AAD	AB%ED	CALC Mode Editor
16AAE - 17A32	FH%TFM	TRANSFORM Execution
17A33 - 17BC6	MN%FRP	FREE/CLAIM PORT
17BC7 - 17DBB	SB%RD	READ Statement Execute
17DBC - 18C57	SB%IO	I/O Execution Routines
18C58 - 18D71	MN%LCK	LOCK Module
18D72 - 1A3D3	SC%SUB	SUBprogram And DEF FN
1A3D4 - 1A57B	SB%EXC	Miscellaneous Execution Routines
1A57C - 1AAA0	SC%REN	RENUMBER Module
1AAA1 - 1AD6D	SG%KEY	Key Routines
1AD6E - 1AF00	SB%VAL	VAL Function
1AF01 - 1B01E	MN%GSB	External GOSUB Module
1B01F - 1B445	AB%UTL	Miscellaneous Utility Routines
1B446 - 1BAA3	MB%IMG	IMAGE Parse Routines
1BAA4 - 1C3C4	MB%USG	USING Execution Routines
1C3C5 - 1C4ED	SB%GPH	Graphics Control Routines
1C4EE - 1C878	SG%POK	POKE, PEEK$, and ADDR$
1C879 - 1DBA5	MN%CD	Card Reader Module
1DBA6 - 1DBA7	JP%ZER	Zero File
1DBA8 - 1DBA9	TI%REV	HP-71 Revision Number
1DBAA - 1DBFE	TI%FX4	ROM 4 Fix Module
1DCCE - 1DDO3	TI%TS	Timestamp File
1DD04 - 1E536	JP%TAB	Lexical Analyzer Tables--ID=01
1E537 - 1F3DF	SB%TAB	Lexical Analyzer Tables--ID=00
1F3E0 - 1FD23	TI%ERM	Mainframe Message Table
1FD24 - 1FE73	SB%KCM	Keycode Map
1FE74 - 1FF2D	AB%LXT	Lextype Table
1FF2E - 1FFFF	SC%TAB	File Type Table
2E100 - 2F9E6	SB%RAM	System RAM Declarations
No address	TI%EQU	Equate File

LIST OF MODULES SORTED BY MODULE NAME	CHAPTER 3

Module	Address Range	Title
ABZASN	0F5DA - 0FA46	Assignment Statement Controller
ABZBLD	1632C - 16671	CALC Mode Decompiler
ABZCLC	1551F - 1632B	CALC Mode
ABZED	16672 - 16AAD	CALC Mode Editor
ABZEXP	0F178 - 0F5D9	Expression Execution Controller
ABZFCN	0B639 - 0C326	Functions
ABZLEX	048E6 - 04F5D	Lexical Analyzer
ABZLXT	1FE74 - 1FF2D	Lextype Table
ABZREG	0ABF8 - 0B638	Variable Creation Routines
ABZUTL	1B01F - 1B445	Miscellaneous Utility Routines
FHZTFM	16AAE - 17A32	TRANSFORM Execution
JPZEXC	07DDC - 08647	Program Execution Routines
JPZMEM	0FF5F - 10191	Program Edit
JPZPOL	1231B - 12549	POLL Routine
JPZPR1	025A0 - 02C25	Line Parse Driver - Part 1
JPZPR2	02C26 - 033D5	Parse Routines - Part 2
JPZPR3	033D6 - 03FD4	Parse Routines - Part 3
JPZSYS	06F2F - 07DDB	System Commands, RUN Loop
JPZTAB	1DD04 - 1E536	Lexical Analyzer Tables--ID=01
JPZZER	0AA81 - 0AA82	Zero File
JPZZER	153A9 - 153AA	Zero File
JPZZER	1DBA6 - 1DBA7	Zero File
JTZMTH	0C327 - 0D434	Math Routines - Part 1
MBZIMG	1B446 - 1BAA3	IMAGE Parse Routines
MBZROM	02484 - 0259F	HP-71 ROM Test
MBZUSG	1BAA4 - 1C3C4	USING Execution Routines
MNZBP	0E9F1 - 0EC6D	BEEP Module
MNZCD	1C879 - 1DBA5	Card Reader Module
MNZCNF	10192 - 11052	Configuration Module
MNZED	14C80 - 153A8	Character Editor
MNZFRP	17A33 - 17BC6	FREE/CLAIM PORT
MNZGSB	1AF01 - 1B01E	External GOSUB Module
MNZLCK	18C58 - 18D71	LOCK
MNZTM	1254A - 13448	Clock System
MNZUTL	0EC6E - 0F177	Miscellaneous Utilities
PMZFLG	13449 - 138B4	Flags, Traps, and Modes
PMZSTA	0DE2F - 0E9F0	Statistics
SBZBIT	00787 - 00C9A	ASCII Tables (Rounded Letters)
SBZCMD	01627 - 0188F	Command Stack Utilities
SBZDSP	01890 - 02483	Display Driver
SBZDVR	00000 - 00784	Main Driver

Module	Address Range	Title
SB%EXC	1A3D4 - 1A57B	Miscellaneous Execution Routines
SB%EXD	05922 - 0624F	Expression Decompile
SB%EXP	03F05 - 048E5	Expression Parser
SB%FCN	0A92B - 0AA80	Function Definitions
SB%FGT	00C9B - 00CF6	F & G Shift Table
SB%GPH	1C3C5 - 1C4ED	Graphics Control Routines
SB%IO	17DBC - 18C57	I/O Execution Routines
SB%KCM	1FD24 - 1FE73	Keycode Map
SB%KEY	00CF7 - 01130	Keyboard Driver (14 Column Keyboard)
SB%RAM	2E100 - 2F9E6	System RAM Declarations
SB%RD	17BC7 - 17DBB	READ Statement Execute
SB%TAB	1E537 - 1F3DF	Lexical Analyzer Tables--ID=00
SB%VAL	1AD6E - 1AFQ0	VAL Function
SC%DAT	138B5 - 14C7F	Store & Retrieve Data - PRINT#/READ#
SC%FIL	11053 - 1231A	File Utility & System Buffer Routines
SC%REN	1A57C - 1AAA0	RENUMBER Module
SC%SUB	18D72 - 1A3D3	SUBprogram And DEF FN
SC%TAB	1FF2E - 1FFFF	File Type Table
SC%TRC	0FA47 - 0FF5E	TRACE Module
SG%EXC	08648 - 09392	Miscellaneous Execution Routines
SG%FXQ	09B76 - 0A92A	File Execution Routines
SG%KEY	1AAA1 - 1AD6D	Key Routines
SG%LDC	04F5E - 05921	Line Decompile
SG%POK	1C4EE - 1C878	POKE, PEEK$, and ADDR$
SG%SYS	063A0 - 06F2E	System Commands: DELETE, AUTO, and CAT
SM%MTH	0D435 - 0DE2E	Math Routines - Part 2
TI%EQU	No address	Equate File
TI%ERD	09393 - 09B75	Error Message Driver
TI%ERM	1F3E0 - 1FD23	Mainframe Message Table
TI%FX1	06250 - 062B1	ROM 1 Fix Module
TI%FX2	0AA85 - 0AACE	ROM 2 Fix Module
TI%FX3	153AC	ROM 3 Fix Module
TI%FX4	1DBAA - 1DBFE	ROM 4 Fix Module
TI%REV	00785 - 00786	HP-71 Revision Number
TI%REV	0AA83 - 0AA84	HP-71 Revision Number
TI%REV	153AB - 153AC	HP-71 Revision Number
TI%REV	1DBA8 - 1DBA9	HP-71 Revision Number
TI%TS	1DCCE - 1DD03	Timestamp File
TI%UTL	01131 - 01626	Command Execution Utilities

```
**********************************************************************************
**                                                                              **
**   77777      1            1      BBBB     BBBB     BBBB     BBBB      **
**      7      11           11      B   B    B   B    B   B    B   B     **
**     7        1            1      B   B    B   B    B   B    B   B     **
**    7         1    ===     1      BBBB     BBBB     BBBB     BBBB      **
**   7          1    ===     1      B   B    B   B    B   B    B   B     **
**   7          1            1      B   B    B   B    B   B    B   B     **
**   7        111          111      BBBB     BBBB     BBBB     BBBB      **
**                                                                              **
**********************************************************************************
SLOAD Rev. 2309/Ver. 1.40

Output module:
TIXR6!:TI:MS::-1    Start=00000 End=1FFFF Length=20000 Syms=2960 Refs=4653
      Date=Fri Dec 30, 1983   5:23 am  Title=HP-71 ROM Release 1BBBB

Source modules:
SB%DVR::MS          Start=00000 End=00784 Length=00785
      Date=Fri Dec 30, 1983   4:11 am  Title=Main Driver <831216.1559>

TIXREV::MS          Start=00785 End=00786 Length=00002
      Date=Fri Dec 30, 1983   5:20 am  Title=HP-71 Revision Number

SB%BIT::MS          Start=00787 End=00C9A Length=00514
      Date=Fri Dec 30, 1983   4:06 am  Title=ASCII Tables (Rounded Letters)

SB%FGT::MS          Start=00C9B End=00CF6 Length=0005C
      Date=Fri Dec 30, 1983   4:21 am  Title=F & G Shift Table<831212.1206>

SB%KEY::MS          Start=00CF7 End=01130 Length=0043A
      Date=Fri Dec 30, 1983   4:26 am  Title=Keyboard Driver (14 Column Keyboard)<83

TIXUTL::MS          Start=01131 End=01626 Length=004F6
      Date=Fri Dec 30, 1983   5:20 am  Title=Command Execution Utilities   <831212.1

SB%CMD::MS          Start=01627 End=0188F Length=00269
      Date=Fri Dec 30, 1983   4:07 am  Title=Command Stack Utilities<831212.1206>

SB%DSP::MS          Start=01890 End=02483 Length=00BF4
      Date=Fri Dec 30, 1983   4:08 am  Title=Display Driver <831216.1505>

MB%ROM::MS          Start=02484 End=0259F Length=0011C
      Date=Fri Dec 30, 1983   3:30 am  Title=HP-71 Rom Test <831212.1205>

JP%PR1::MS          Start=025A0 End=02C25 Length=00686
      Date=Fri Dec 30, 1983   2:58 am  Title=Line Parse Driver - Part 1 <831212.1516

JP%PR2::MS          Start=02C26 End=033D5 Length=007B0
      Date=Fri Dec 30, 1983   3:01 am  Title=Parse Routines - Part 2<831212.1204>

JP%PR3::MS          Start=033D6 End=03FD4 Length=00BFF
      Date=Fri Dec 30, 1983   3:04 am  Title=Parse Routines - Part 3<831212.1204>

SB%EXP::MS          Start=03FD5 End=048E5 Length=00911
      Date=Fri Dec 30, 1983   4:18 am  Title=Expression Parser<831212.1206>

AB%LEX::MS          Start=048E6 End=04F5D Length=00678
      Date=Fri Dec 30, 1983   2:38 am  Title=Lexical Analyzer <831212.1512>
```

```
SG%LDC::MS          Start=04F5E End=05921 Length=009C4
     Date=Fri Dec 30, 1983   5:00 am  Title=Line Decompile <831216.1615>

SB%EXD::MS          Start=05922 End=0624F Length=0092E
     Date=Fri Dec 30, 1983   4:15 am  Title=Expression Decompile <831212.1206>

TI%FX1::MS          Start=06250 End=062B1 Length=00062
     Date=Fri Dec 30, 1983   5:19 am  Title=ROM 1 Fix Module

SG%SYS::MS          Start=063A0 End=06F2E Length=00B8F
     Date=Fri Dec 30, 1983   5:06 am  Title=System Commands: DELETE, AUTO, and CAT

JP%SYS::MS          Start=06F2F End=07DDB Length=00EAD
     Date=Fri Dec 30, 1983   3:10 am  Title=System Commands, RUN Loop<831212.1204>

JP%EXC::MS          Start=07DDC End=08647 Length=0086C
     Date=Fri Dec 30, 1983   2:50 am  Title=Program Execution Routines <831212.1515

SG%EXC::MS          Start=08648 End=09392 Length=00D4B
     Date=Fri Dec 30, 1983   4:49 am  Title=Miscellaneous Execution Routines<831216

TI%ERD::MS          Start=09393 End=09B75 Length=007E3
     Date=Fri Dec 30, 1983   5:13 am  Title=Error Message Driver <831216.1627>

SG%FXQ::MS          Start=09B76 End=0A92A Length=00DB5
     Date=Fri Dec 30, 1983   4:54 am  Title=File Execution Routines<831216.1614>

SB%FCN::MS          Start=0A92B End=0AA80 Length=00156
     Date=Fri Dec 30, 1983   4:20 am  Title=Function Definitions <831212.1206>

JP%ZER::MS          Start=0AA81 End=0AA82 Length=00002
     Date=Fri Dec 30, 1983   3:18 am  Title=Zero File - End of chain

TI%REV::MS          Start=0AA83 End=0AA84 Length=00002
     Date=Fri Dec 30, 1983   5:20 am  Title=HP-71 Revision Number

TI%FX2::MS          Start=0AA85 End=0AACE Length=0004A
     Date=Fri Dec 30, 1983   5:19 am  Title=ROM 2 Fix Module

AB%REG::MS          Start=0ABF8 End=0B638 Length=00A41
     Date=Fri Dec 30, 1983   2:39 am  Title=Variable Creation Routines <831216.1818

AB%FCN::MS          Start=0B639 End=0C326 Length=00CEE
     Date=Fri Dec 30, 1983   2:34 am  Title=Functions <831212.1512>

JT%MTH::MS          Start=0C327 End=0D434 Length=0110E
     Date=Fri Dec 30, 1983   3:18 am  Title=Math Routines - Part 1 <831213.1007>

SM%MTH::MS          Start=0D435 End=0DE2E Length=009FA
     Date=Fri Dec 30, 1983   5:10 am  Title=Math Routines - Part 2 <831216.1618>

PM%STA::MS          Start=0DE2F End=0E9F0 Length=00BC2
     Date=Fri Dec 30, 1983   4:01 am  Title=Statistics <831216.1457>

MN%BP::MS           Start=0E9F1 End=0EC6D Length=0027D
     Date=Fri Dec 30, 1983   3:34 am  Title=BEEP Module <831212.1529>

MN%UTL::MS          Start=0EC6E End=0F177 Length=0050A
     Date=Fri Dec 30, 1983   3:57 am  Title=Miscellaneous Utilities<831216.1456>
```

```
AB%EXP::MS          Start=0F178 End=0F5D9 Length=00462
      Date=Fri Dec 30, 1983   2:33 am  Title=Expression Execution Controller<831212.

AB%ASN::MS          Start=0F5DA End=0FA46 Length=0046D
      Date=Fri Dec 30, 1983   2:27 am  Title=Assignment Statement Controller<831216.

SC%TRC::MS          Start=0FA47 End=0FF5E Length=00518
      Date=Fri Dec 30, 1983   4:48 am  Title=TRACE Module <831216.1824>

JP%MEM::MS          Start=0FF5F End=10191 Length=00233
      Date=Fri Dec 30, 1983   2:55 am  Title=Program Edit <831212.1204>

MN%CNF::MS          Start=10192 End=11052 Length=00EC1
      Date=Fri Dec 30, 1983   3:42 am  Title=Configuration Module <831213.1255>

SC%FIL::MS          Start=11053 End=1231A Length=012C8
      Date=Fri Dec 30, 1983   4:34 am  Title=File Utility & System Buffer Routines<8

JP%POL::MS          Start=1231B End=12549 Length=0022F
      Date=Fri Dec 30, 1983   2:56 am  Title=POLL Routine <831212.1204>

MN%TM::MS           Start=1254A End=13448 Length=00EFF
      Date=Fri Dec 30, 1983   3:51 am  Title=Clock System <831228.1028>

PM%FLG::MS          Start=13449 End=138B4 Length=0046C
      Date=Fri Dec 30, 1983   3:59 am  Title=Flags, Traps, and Modes<831216.1456>

SC%DAT::MS          Start=138B5 End=14C7F Length=013CB
      Date=Fri Dec 30, 1983   4:28 am  Title=Store & Retrieve Data - PRINT#/READ#<83

MN%ED::MS           Start=14C80 End=153A8 Length=00729
      Date=Fri Dec 30, 1983   3:47 am  Title=Character Editor <831213.1314>

JP%ZER::MS          Start=153A9 End=153AA Length=00002
      Date=Fri Dec 30, 1983   3:18 am  Title=Zero File - End of chain

TI%REV::MS          Start=153AB End=153AC Length=00002
      Date=Fri Dec 30, 1983   5:20 am  Title=HP-71 Revision Number

TI%FX3::MS         Module Contains No Code
      Date=Fri Dec 30, 1983   5:20 am  Title=ROM 3 Fix Module

AB%CLC::MS          Start=1551F End=1632B Length=00E0D
      Date=Fri Dec 30, 1983   2:29 am  Title=CALC Mode <831228.1215>

AB%BLD::MS          Start=1632C End=16671 Length=00346
      Date=Fri Dec 30, 1983   2:29 am  Title=CALC Mode Decompiler <831212.1509>

AB%ED::MS           Start=16672 End=16AAD Length=0043C
      Date=Fri Dec 30, 1983   2:32 am  Title=CALC Mode Editor <831212.1510>

FH%TFM::MS          Start=16AAE End=17A32 Length=00F85
      Date=Fri Dec 30, 1983   2:44 am  Title=TRANSFORM Execution <831212.1203>

MN%FRP::MS          Start=17A33 End=17BC6 Length=00194
      Date=Fri Dec 30, 1983   3:49 am  Title=FREE/CLAIM PORT <831212.1205>

SB%RD::MS           Start=17BC7 End=17DBB Length=001F5
      Date=Fri Dec 30, 1983   4:27 am  Title=READ Statement Execute <831216.1605>
```

```
SB%IO::MS           Start=17DBC End=18C57 Length=00E9C
      Date=Fri Dec 30, 1983   4:21 am  Title=I/O Execution Routines <831228.1849>

MN%LCK::MS          Start=18C58 End=18D71 Length=0011A
      Date=Fri Dec 30, 1983   3:50 am  Title=LOCK Module <831216.1454>

SC%SUB::MS          Start=18D72 End=1A3D3 Length=01662
      Date=Fri Dec 30, 1983   4:42 am  Title=SUBprogram and DEF FN <831216.1610>

SB%EXC::MS          Start=1A3D4 End=1A57B Length=001A8
      Date=Fri Dec 30, 1983   4:15 am  Title=Miscellaneous Execution Routines <83121

SC%REN::MS          Start=1A57C End=1AAA0 Length=00525
      Date=Fri Dec 30, 1983   4:41 am  Title=RENUMBER Module <831216.1609>

SG%KEY::MS          Start=1AAA1 End=1AD6D Length=002CD
      Date=Fri Dec 30, 1983   4:59 am  Title=Key Routines <831212.1207>

SB%VAL::MS          Start=1AD6E End=1AF00 Length=00193
      Date=Fri Dec 30, 1983   4:28 am  Title=VAL Function <831212.1206>

MN%GSB::MS          Start=1AF01 End=1B01E Length=0011E
      Date=Fri Dec 30, 1983   3:50 am  Title=External GOSUB Module<831216.1453>

AB%UTL::MS          Start=1B01F End=1B445 Length=00427
      Date=Fri Dec 30, 1983   2:42 am  Title=Miscellaneous Utility Routines <831212.

MB%IMG::MS          Start=1B446 End=1BAA3 Length=0065E
      Date=Fri Dec 30, 1983   3:24 am  Title=IMAGE Parse Routines <831212.1518>

MB%USG::MS          Start=1BAA4 End=1C3C4 Length=00921
      Date=Fri Dec 30, 1983   3:30 am  Title=USING Execution Routines <831212.1519>

SB%GPH::MS          Start=1C3C5 End=1C4ED Length=00129
      Date=Fri Dec 30, 1983   4:21 am  Title=Graphics Control Routines

SG%POK::MS          Start=1C4EE End=1C878 Length=0038B
      Date=Fri Dec 30, 1983   5:04 am  Title=POKE, PEEK$, and ADDR$ <831216.1616>

MN%CD::MS           Start=1C879 End=1DBA5 Length=0132D
      Date=Fri Dec 30, 1983   3:35 am  Title=Card Reader Module <831212.1617>

JP%ZER::MS          Start=1DBA6 End=1DBA7 Length=00002
      Date=Fri Dec 30, 1983   3:18 am  Title=Zero File - End of chain

TI%REV::MS          Start=1DBA8 End=1DBA9 Length=00002
      Date=Fri Dec 30, 1983   5:20 am  Title=HP-71 Revision Number

TI%FX4::MS          Start=1DBAA End=1DBFE Length=00055
      Date=Fri Dec 30, 1983   5:20 am  Title=ROM 4 Fix Module

TI%TS::MS           Start=1DCCE End=1DD03 Length=00036
      Date=Fri Dec 30, 1983   5:20 am  Title=Timestamp File

JP%TAB::MS          Start=1DD04 End=1E536 Length=00833
      Date=Fri Dec 30, 1983   2:24 am  Title=Lexical Analyzer Tables--ID=01

SB%TAB::MS          Start=1E537 End=1F3DF Length=00EA9
      Date=Fri Dec 30, 1983   2:21 am  Title=Lexical Analyzer Tables--ID=00
```

```
TIXERM::MS          Start=1F3E0 End=1FD23 Length=00944
     Date=Fri Dec 30, 1983   5:17 am  Title=Mainframe Message Table

SBXKCM::MS          Start=1FD24 End=1FE73 Length=00150
     Date=Fri Dec 30, 1983   2:27 am  Title=Keycode Map<831212.1206>

RBXLXT::MS          Start=1FE74 End=1FF2D Length=000BA
     Date=Fri Dec 30, 1983   2:39 am  Title=Lextype Table <831212.1512>

SCXTAB::MS          Start=1FF2E End=1FFFF Length=000D2
     Date=Fri Dec 30, 1983   4:47 am  Title=File Type Table

SBXRAM::MS          Module Contains No Code
     Date=Fri Dec 30, 1983   2:20 am  Title=System RAM Declarations<831212.1206>

TIXEQU::MS          Module Contains No Code
     Date=Fri Dec 30, 1983   2:25 am  Title=Equate File <831228.1851>
```

Saturn Long Cross Reference Listing

```
!CK     =  02E0C JP%PR2        - 0277D JP%PR1(001DD) Type=1.1 Nibs=3 Dist=0068F
                               + 0294F JP%PR1(003AF) Type=1.1 Nibs=3 Dist=004BD
                               + 034AE JP%PR3(000D8) Type=1.1 Nibs=3 Dist=006A2
!CK2    =  02E13 JP%PR2        - 0382B JP%PR3(00455) Type=1.1 Nibs=4 Dist=00A18
!CK3    =  02E1F JP%PR2        - 02844 JP%PR1(002A4) Type=1.1 Nibs=3 Dist=005DB

#CK     =  03356 JP%PR2        - 02B5D JP%PR1(005BD) Type=1.1 Nibs=4 Dist=007F9
                               + 033DA JP%PR3(00004) Type=1.1 Nibs=3 Dist=00084
                               + 037BE JP%PR3(003E8) Type=1.1 Nibs=3 Dist=00468
                               + 038DE JP%PR3(00508) Type=1.1 Nibs=3 Dist=00588
                               + 03ADC JP%PR3(00706) Type=1.1 Nibs=3 Dist=00786

-CHAR   =  1527D MN%ED:        -
-LINE   =  15275 MN%ED:        - 18AF4 SB%IO:(00D38) Type=1.1 Nibs=4 Dist=0387F

/LINE#  =  2F8E4 FH%TFM        - 172BA FH%TFM(0080C) Type=0.0 Nibs=2

1/X15   =  0C33E JT%MTH        - 0D8F7 SM%MTH(004C2) Type=1.1 Nibs=4 Dist=015B9
                               + 0D9EC SM%MTH(005B7) Type=1.1 Nibs=4 Dist=016AE
                               + 0DCD8 SM%MTH(008A3) Type=1.1 Nibs=4 Dist=0199A
1/X15S  =  0C344 JT%MTH        -

?PRFI+  =  17380 FH%TFM        -
?PRFIL  =  1737E FH%TFM        -

A-MULT  =  1B349 AB%UTL        - 117E8 SC%FIL(00795) Type=0.1 Nibs=5
                               + 14791 SC%DAT(00EDC) Type=1.1 Nibs=4 Dist=06BB8
A=CUR   =  020AA SB%DSP        - 095C0 TI%ERD(0022D) Type=1.1 Nibs=4 Dist=07516
ABS     =  0B909 AB%FCN        - 1EB54 SB%TAB(0061D) Type=1.2 Nibs=5 Dist=1324B
ACBAT?  =  0EF29 MN%UTL        - 00658 SB%DVR(00658) Type=0.1 Nibs=5
                               + 1CFE4 MN%CD:(0076B) Type=0.1 Nibs=5
ACBTSR  =  0EF17 MN%UTL        - 00723 SB%DVR(00723) Type=0.1 Nibs=5
ACCEPT  =  0450F SB%EXP        -
ACOS    =  0BDE9 AB%FCN        - 1DD6F JP%TAB(0006B) Type=1.2 Nibs=5 Dist=11F86
                               + 1EB0C SB%TAB(005D5) Type=1.2 Nibs=5 Dist=12D23
ACOS12  =  0DBD3 SM%MTH        - 0BDF2 AB%FCN(007B9) Type=1.1 Nibs=4 Dist=01DE1
ACOS15  =  0DBD7 SM%MTH        -
ACTIVE  =  2F5A8 SB%RAM        - 07D3D JP%SYS(00E0E) Type=0.0 Nibs=5
                               + 0B2A2 AB%REG(006AA) Type=0.0 Nibs=5
                               + 197D7 SC%SUB(00A65) Type=0.0 Nibs=5
ACTTMR  =  131A9 MN%TM:        - 07EDC JP%EXC(00100) Type=0.1 Nibs=5
AD15M   =  0C366 JT%MTH        -
AD15S   =  0E19D PM%STA        -
AD15s   =  0C369 JT%MTH        - 0DDC8 SM%MTH(00993) Type=1.1 Nibs=4 Dist=01A5F
                               + 0DDD7 SM%MTH(009A2) Type=1.1 Nibs=4 Dist=01A6E
                               + 0E1A5 PM%STA(00376) Type=1.1 Nibs=4 Dist=01E3C
AD2-12  =  0C35F JT%MTH        -
AD2-15  =  0C363 JT%MTH        -
ADD     =  0DE72 PM%STA        - 1ED1F SB%TAB(007E8) Type=1.2 Nibs=5 Dist=10EAD
ADDDC   =  05470 SG%LDC        - 0DE68 PM%STA(00039) Type=1.2 Nibs=5 Dist=089F8
ADDF    =  0C372 JT%MTH        - 0D6FC SM%MTH(002C7) Type=1.0 Nibs=4 Dist=0138A
ADDONE  =  0C330 JT%MTH        - 0D9E0 SM%MTH(005AB) Type=1.1 Nibs=4 Dist=016B0
                               + 0DC9D SM%MTH(00868) Type=1.1 Nibs=4 Dist=0196D
                               + 0E889 PM%STA(00A5A) Type=1.1 Nibs=4 Dist=02559
ADDP    =  03A03 JP%PR3        - 0DE6D PM%STA(0003E) Type=1.2 Nibs=5 Dist=0A46A
ADDR$   =  1C801 SG%POK        - 1DD78 JP%TAB(00074) Type=1.2 Nibs=5 Dist=01577
ADDRSS  =  0F527 AB%EXP        - 0ADDF AB%REG(001E7) Type=1.1 Nibs=4 Dist=04748
                               + 0B0CB AB%REG(004D3) Type=1.1 Nibs=4 Dist=0445C
```

```
                                + 0BADF AB%FCN(004A6) Type=1.1 Nibs=4 Dist=03A48
                                + 0BBB4 AB%FCN(0057B) Type=1.1 Nibs=4 Dist=03973
                                + 15FB6 AB%CLC(00A97) Type=1.1 Nibs=4 Dist=06A8F
                                + 1DBEB TI%FX4(00041) Type=0.1 Nibs=5
ADHDJ  =  1ACEE SG%KEY          - 1A579 SB%EXC(001A5) Type=1.0 Nibs=3 Dist=00775
                                + 1ADA3 SB%VAL(00035) Type=1.1 Nibs=3 Dist=000B5
ADHEAD =  181B7 SB%IO:          - 0AA45 SB%FCN(0011A) Type=0.1 Nibs=5
                                + 17D60 SB%RD:(00199) Type=1.1 Nibs=3 Dist=00457
                                + 1ACF0 SG%KEY(0024F) Type=1.0 Nibs=4 Dist=02B39
ADJA   =  1289A MN%TM:          -
ADJAAA =  12D08 MN%TM:          - 1DD81 JP%TAB(0007D) Type=1.2 Nibs=5 Dist=0B079
ADJN   =  12825 MN%TM:          -
ADJNNN =  12D2C MN%TM:          - 1DD8A JP%TAB(00086) Type=1.2 Nibs=5 Dist=0B05E
ADJPTR =  0B527 AB%REG          - 194BA SC%SUB(00748) Type=0.1 Nibs=5
ADJREF =  013A9 TI%UTL          -
ADRS40 =  0F52B AB%EXP          - 0AE5A AB%REG(00262) Type=1.1 Nibs=4 Dist=046D1
                                + 0AFCF AB%REG(003D7) Type=1.1 Nibs=4 Dist=0455C
                                + 0FB9E SC%TRC(00157) Type=1.1 Nibs=3 Dist=00673
ADRS50 =  0F551 AB%EXP          - 0E426 PM%STA(005F7) Type=1.1 Nibs=4 Dist=0112B
                                + 0E568 PM%STA(00739) Type=1.1 Nibs=4 Dist=00FE9
ADRS80 =  0F567 AB%EXP          - 199A9 SC%SUB(00C37) Type=0.1 Nibs=5
ADRSUB =  0F4CF AB%EXP          - 1A087 SC%SUB(01315) Type=0.1 Nibs=5
AF     =  12DEC MN%TM:          - 1DD93 JP%TAB(0008F) Type=1.2 Nibs=5 Dist=0AFA7
AJDEST =  0B3BE AB%REG          - 0F8EB AB%ASN(00311) Type=1.1 Nibs=4 Dist=0452D
ALINFO =  01531 TI%UTL          - 06FE7 JP%SYS(000B8) Type=1.1 Nibs=4 Dist=05AB6
                                + 081CF JP%EXC(003F3) Type=1.1 Nibs=4 Dist=06C9E
                                + 16ABA FH%TFM(0000C) Type=0.1 Nibs=5
ALLDUN =  04BEF AB%LEX          -
ALMSRV =  1257D MN%TM:          - 005F9 SB%DVR(005F9) Type=0.1 Nibs=5
                                + 00747 SB%DVR(00747) Type=0.1 Nibs=5
                                + 0215A SB%DSP(008CA) Type=0.1 Nibs=5
                                + 07593 JP%SYS(00664) Type=0.1 Nibs=5
                                + 0ED76 MN%UTL(00108) Type=1.1 Nibs=4 Dist=03807
                                + 14E5B MN%ED:(001DB) Type=1.1 Nibs=4 Dist=028DE
ALRM1  =  2F719 SB%RAM          - 1295A MN%TM:(00410) Type=0.0 Nibs=5
                                + 131C4 MN%TM:(00C7A) Type=0.0 Nibs=5 Offset=    -12
ALRM2  =  2F725 SB%RAM          -
ALRM3  =  2F731 SB%RAM          -
ALRM4  =  2F73D SB%RAM          -
ALRM5  =  2F749 SB%RAM          -
ALRM6  =  2F755 SB%RAM          - 1270D MN%TM:(001C3) Type=0.0 Nibs=5
AND    =  0B877 AB%FCN          - 1EA85 SB%TAB(0054E) Type=1.2 Nibs=5 Dist=1320E
ANGLE  =  0BE12 AB%FCN          - 1DD9C JP%TAB(00098) Type=1.2 Nibs=5 Dist=11F8A
ANN1.5 =  2E101 SB%RAM          -
ANNAD1 =  2E100 SB%RAM          - 01905 SB%DSP(00075) Type=0.0 Nibs=5
                                + 135CA PM%FLG(00181) Type=0.0 Nibs=4 Offset=     1
ANNAD2 =  2E102 SB%RAM          - 000C6 SB%DVR(000C6) Type=0.0 Nibs=4
                                + 005C9 SB%DVR(005C9) Type=0.0 Nibs=4
                                + 00E35 SB%KEY(0013E) Type=0.0 Nibs=5
                                + 00E5D SB%KEY(00166) Type=0.0 Nibs=5
                                + 00FE7 SB%KEY(002F0) Type=0.0 Nibs=4
                                + 1359A PM%FLG(00151) Type=0.0 Nibs=4 Offset=     1
ANNAD3 =  2E34C SB%RAM          - 135AC PM%FLG(00163) Type=0.0 Nibs=4
ANNAD4 =  2E34E SB%RAM          - 01965 SB%DSP(000D5) Type=0.0 Nibs=4
                                + 1358C PM%FLG(00143) Type=0.0 Nibs=4
ARANGE =  045D1 SB%EXP          - 03744 JP%PR3(0036E) Type=1.1 Nibs=4 Dist=00E8D
                                + 03F88 JP%PR3(00BB2) Type=1.1 Nibs=3 Dist=00649
                                + 053BE SG%LDC(00460) Type=1.1 Nibs=4 Dist=00DED
                                + 08983 SG%EXC(0033B) Type=1.1 Nibs=4 Dist=043B2
                                + 151A6 MN%ED:(00526) Type=0.1 Nibs=5
```

```
ARG12  =  0D67B SM%MTH        - 0BE1B AB%FCN(007E2) Type=1.1 Nibs=4 Dist=01860
ARG15  =  0D67F SM%MTH        -
ARGCNT =  16137 AB%CLC        - 167C6 AB%ED:(00154) Type=1.1 Nibs=3 Dist=0068F
ARGERR =  0BF19 AB%FCN        - 06BEA SG%SYS(0084A) Type=1.0 Nibs=4 Dist=0532F
                              + 080BC JP%EXC(002E0) Type=1.0 Nibs=4 Dist=03E5D
                              + 08DC4 SG%EXC(0077C) Type=1.0 Nibs=4 Dist=03155
                              + 09F1D SG%FXQ(003A7) Type=1.0 Nibs=4 Dist=01FFC
                              + 0B067 AB%REG(0046F) Type=1.0 Nibs=4 Dist=00EB2
                              + 0E922 PM%STA(00AF3) Type=1.0 Nibs=4 Dist=02A09
                              + 11453 SC%FIL(00400) Type=1.0 Nibs=4 Dist=0553A
                              + 12D4A MN%TM:(00800) Type=1.0 Nibs=4 Dist=06E31
ARGF   =  0D6A4 SM%MTH        -
ARGPR+ =  0E8EB PM%STA        -
ARGPRP =  0E8EF PM%STA        -
ARGST- =  0E910 PM%STA        - 136CD PM%FLG(00284) Type=1.1 Nibs=4 Dist=04DBD
ARGSTA =  0E90C PM%STA        - 07E7E JP%EXC(000A2) Type=1.1 Nibs=4 Dist=06A8E
                              + 080A8 JP%EXC(002CC) Type=1.1 Nibs=4 Dist=06864
                              + 09EA6 SG%FXQ(00330) Type=1.1 Nibs=4 Dist=04A66
                              + 0EE8C MN%UTL(0021E) Type=1.1 Nibs=3 Dist=00580
                              + 12CF4 MN%TM:(007AA) Type=1.0 Nibs=4 Dist=043E8
ARITH  =  061E0 SB%EXD        - 16483 AB%BLD(00157) Type=0.1 Nibs=5
ARRAY  =  0BADD AB%FCN        - 04BDB AB%LEX(002F5) Type=0.0 Nibs=5
                              + 1EA07 SB%TAB(004D0) Type=1.2 Nibs=5 Dist=12F2A
ARRYCK =  0366A JP%PR3        - 0329C JP%PR2(00676) Type=1.1 Nibs=3 Dist=003CE
                              + 032CE JP%PR2(006A8) Type=1.1 Nibs=3 Dist=0039C
ARYDC  =  05178 SG%LDC        -
ARYELM =  0B5A7 AB%REG        - 147A8 SC%DAT(00EF3) Type=0.1 Nibs=5
ARYSIZ =  0B61B AB%REG        - 1984F SC%SUB(00ADD) Type=0.1 Nibs=5
ASCO2  =  0515A SG%LDC        -
ASCICK =  0514E SG%LDC        -
ASCII  =  0079B SB%BIT        - 019F7 SB%DSP(00167) Type=0.0 Nibs=5
ASIN   =  0BDCF AB%FCN        - 1DDA5 JP%TAB(000A1) Type=1.2 Nibs=5 Dist=11FD6
                              + 1EB03 SB%TAB(005CC) Type=1.2 Nibs=5 Dist=12D34
ASIN12 =  0DBC8 SM%MTH        - 0BDD8 AB%FCN(0079F) Type=1.1 Nibs=4 Dist=01DF0
ASIN15 =  0DBCC SM%MTH        -
ASLW3  =  0ED21 MN%UTL        - 101D6 MN%CNF(00044) Type=1.1 Nibs=4 Dist=014B5
ASLW4  =  0ED1E MN%UTL        - 1336E MN%TM:(00E24) Type=1.1 Nibs=4 Dist=04650
ASLW5  =  0ED1B MN%UTL        - 10F94 MN%CNF(00E02) Type=1.1 Nibs=4 Dist=02279
                              + 14E9A MN%ED:(0021A) Type=1.1 Nibs=4 Dist=0617F
                              + 1D20D MN%CD:(00994) Type=0.1 Nibs=5
ASNMNT =  0F5E0 AB%ASN        - 074C1 JP%SYS(00592) Type=0.1 Nibs=5
                              + 086B9 SG%EXC(00071) Type=1.1 Nibs=4 Dist=06F27
ASNMP  =  0314E JP%PR2        - 02B25 JP%PR1(00585) Type=1.0 Nibs=3 Dist=00629
ASNSTO =  0F5ED AB%ASN        - 19A02 SC%SUB(00C90) Type=0.1 Nibs=5
                              + 19E49 SC%SUB(010D7) Type=0.1 Nibs=5
ASRW3  =  0ED10 MN%UTL        - 101E6 MN%CNF(00054) Type=1.1 Nibs=4 Dist=014D6
                              + 12FFB MN%TM:(00AB1) Type=1.1 Nibs=4 Dist=042EB
ASRW4  =  0EDOD MN%UTL        - 1D5D6 MN%CD:(00D5D) Type=0.1 Nibs=5
ASRW5  =  0EDOA MN%UTL        - 1D214 MN%CD:(0099B) Type=0.1 Nibs=5
ASSIGN =  11F33 SC%FIL        - 1DDAE JP%TAB(000AA) Type=1.2 Nibs=5 Dist=0BE7B
ASSNDC =  054C9 SG%LDC        - 11F29 SC%FIL(00ED6) Type=1.2 Nibs=5 Dist=0CA60
ASSNP  =  03ADB JP%PR3        - 11F2E SC%FIL(00EDB) Type=1.2 Nibs=5 Dist=0E453
ATAN   =  0BDFD AB%FCN        - 1DDB7 JP%TAB(000B3) Type=1.2 Nibs=5 Dist=11FBA
                              + 1EB15 SB%TAB(005DE) Type=1.2 Nibs=5 Dist=12D18
ATAN12 =  0DBBA SM%MTH        - 0BE06 AB%FCN(007CD) Type=1.1 Nibs=4 Dist=01DB4
ATAN15 =  0DBBE SM%MTH        -
ATCHK  =  077D7 JP%SYS        - 0963A TI%ERD(002A7) Type=1.0 Nibs=4 Dist=01E63
ATNCLR =  00510 SB%DVR        - 18C53 SB%IO:(00E97) Type=0.1 Nibs=5
ATNDIS =  2F441 SB%RAM        - 00FB8 SB%KEY(002C1) Type=0.0 Nibs=4
ATNFLG =  2F442 SB%RAM        - 00512 SB%DVR(00512) Type=0.0 Nibs=5
```

```
                                + 01DEA SB%DSP(0055A) Type=0.0 Nibs=5
                                + 0246F SB%DSP(00BDF) Type=0.0 Nibs=5
                                + 076AF JP%SYS(00780) Type=0.0 Nibs=5
                                + 0ED81 MN%UTL(00113) Type=0.0 Nibs=5
ATTNTN =  004E3 SB%DVR          -
AUTCLR =  029A7 JP%PR1          - 00271 SB%DVR(00271) Type=1.1 Nibs=4 Dist=02736
                                + 00504 SB%DVR(00504) Type=1.1 Nibs=4 Dist=024A3
                                + 070FA JP%SYS(001CB) Type=1.1 Nibs=4 Dist=04753
                                + 0722E JP%SYS(002FF) Type=1.1 Nibs=4 Dist=04887
                                + 09485 TI%ERD(000F2) Type=1.1 Nibs=4 Dist=06ADE
AUTDEL =  0F0F9 MN%UTL          - 10F17 MN%CNF(00D85) Type=1.0 Nibs=4 Dist=01E1E
AUTINC =  2F6CB SB%RAM          - 029AE JP%PR1(0040E) Type=0.0 Nibs=5
                                + 02BD2 JP%PR1(00632) Type=0.0 Nibs=5
                                + 069BC SG%SYS(0061C) Type=0.0 Nibs=5
                                + 1771A FH%TFM(00C6C) Type=0.0 Nibs=5
AUTLN2 =  069FA SG%SYS          - 073F1 JP%SYS(004C2) Type=1.0 Nibs=4 Dist=009F7
AUTO   =  0699B SG%SYS          - 1EE00 SB%TAB(008C9) Type=1.2 Nibs=5 Dist=18465
AUTOCK =  02BCD JP%PR1          - 0031B SB%DVR(0031B) Type=1.1 Nibs=4 Dist=028B2
                                + 07414 JP%SYS(004E5) Type=1.1 Nibs=4 Dist=04847
AUTOP  =  03C03 JP%PR3          - 06996 SG%SYS(005F6) Type=1.2 Nibs=5 Dist=02D93
AUTXQ7 =  069C9 SG%SYS          - 00324 SB%DVR(00324) Type=1.0 Nibs=4 Dist=066A5
AVE=C  =  18BBB SB%IO:          - 17D81 SB%RD:(001BA) Type=1.1 Nibs=4 Dist=00E3A
AVE=D1 =  18BB8 SB%IO:          - 09F46 SG%FXQ(003D0) Type=0.1 Nibs=5
                                + 1AEEE SB%VAL(00180) Type=1.0 Nibs=4 Dist=02336
                                + 1BAD2 MB%USG(0002E) Type=1.1 Nibs=4 Dist=02F1A
                                + 1C0AA MB%USG(00606) Type=1.1 Nibs=4 Dist=034F2
AVM2DS =  01C01 SB%DSP          - 0665A SG%SYS(002BA) Type=1.1 Nibs=4 Dist=04A59
                                + 0A4CF SG%FXQ(00959) Type=0.1 Nibs=5
AVMEME =  2F599 SB%RAM          - 01291 TI%UTL(00160) Type=0.0 Nibs=5
                                + 012BD TI%UTL(0018C) Type=0.0 Nibs=5
                                + 015EE TI%UTL(004BD) Type=0.0 Nibs=5
                                + 0451D SB%EXP(00548) Type=0.0 Nibs=5
                                + 08939 SG%EXC(002F1) Type=0.0 Nibs=5
                                + 08B6C SG%EXC(00524) Type=0.0 Nibs=5
                                + 08C92 SG%EXC(0064A) Type=0.0 Nibs=2
                                + 08F57 SG%EXC(0090F) Type=0.0 Nibs=5
                                + 08F85 SG%EXC(0093D) Type=0.0 Nibs=5
                                + 09850 TI%ERD(004BD) Type=0.0 Nibs=5
                                + 0BC4E AB%FCN(00615) Type=0.0 Nibs=4
                                + 0CCF9 JT%MTH(009D2) Type=0.0 Nibs=5
                                + 0E1FD PM%STA(003CE) Type=0.0 Nibs=4
                                + 10C57 MN%CNF(00AC5) Type=0.0 Nibs=5
                                + 17AC1 MN%FRP(0008E) Type=0.0 Nibs=4
                                + 18653 SB%IO:(00897) Type=0.0 Nibs=5
                                + 18893 SB%IO:(00AD7) Type=0.0 Nibs=5
                                + 18924 SB%IO:(00B68) Type=0.0 Nibs=5
                                + 189C2 SB%IO:(00C06) Type=0.0 Nibs=5
                                + 18BBD SB%IO:(00E01) Type=0.0 Nibs=5
                                + 18D02 MN%LCK(000AA) Type=0.0 Nibs=5
                                + 1A47B SB%EXC(000A7) Type=0.0 Nibs=5
                                + 1B0FC AB%UTL(000DD) Type=0.0 Nibs=5
AVMEMS =  2F594 SB%RAM          - 0025E SB%DVR(0025E) Type=0.0 Nibs=2
                                + 0129B TI%UTL(0016A) Type=0.0 Nibs=5
                                + 012CE TI%UTL(0019D) Type=0.0 Nibs=5
                                + 01499 TI%UTL(00368) Type=0.0 Nibs=5
                                + 01C03 SB%DSP(00373) Type=0.0 Nibs=5
                                + 04720 SB%EXP(0074B) Type=0.0 Nibs=5
                                + 0475C SB%EXP(00787) Type=0.0 Nibs=5
                                + 09102 SG%EXC(00ABA) Type=0.0 Nibs=5
                                + 09319 SG%EXC(00CD1) Type=0.0 Nibs=5
```

```
                                + 09B2E TI%ERD(0079B) Type=0.0 Nibs=5
                                + 0A167 SG%FXQ(005F1) Type=0.0 Nibs=5
                                + 0A22D SG%FXQ(006B7) Type=0.0 Nibs=5
                                + 0A654 SG%FXQ(00ADE) Type=0.0 Nibs=5
                                + 0A6FD SG%FXQ(00B87) Type=0.0 Nibs=5
                                + 0B41B AB%REG(00823) Type=0.0 Nibs=5
                                + 0BFD7 AB%FCN(0099E) Type=0.0 Nibs=5
                                + 0C1B4 AB%FCN(00B7B) Type=0.0 Nibs=5
                                + 0F0AA MN%UTL(0043C) Type=0.0 Nibs=5
                                + 0F21B AB%EXP(000A3) Type=0.0 Nibs=5
                                + 0F37E AB%EXP(00206) Type=0.0 Nibs=5
                                + 0F41C AB%EXP(002A4) Type=0.0 Nibs=5
                                + 1086D MN%CNF(006DB) Type=0.0 Nibs=5
                                + 109F2 MN%CNF(00860) Type=0.0 Nibs=5
                                + 10CA4 MN%CNF(00B12) Type=0.0 Nibs=5
                                + 13DBB SC%DAT(00506) Type=0.0 Nibs=5
                                + 15828 AB%CLC(00309) Type=0.0 Nibs=5
                                + 15B9D AB%CLC(0067E) Type=0.0 Nibs=5
                                + 15EA1 AB%CLC(00982) Type=0.0 Nibs=5
                                + 15F99 AB%CLC(00A7A) Type=0.0 Nibs=5
                                + 1636B AB%BLD(0003F) Type=0.0 Nibs=5
                                + 163D7 AB%BLD(000AB) Type=0.0 Nibs=5
                                + 163EC AB%BLD(000C0) Type=0.0 Nibs=5
                                + 176A2 FH%TFM(00BF4) Type=0.0 Nibs=5
                                + 187EA SB%IO:(00A2E) Type=0.0 Nibs=5
                                + 19CB8 SC%SUB(00F46) Type=0.0 Nibs=5
                                + 1A465 SB%EXC(00091) Type=0.0 Nibs=5
                                + 1B16A AB%UTL(0014B) Type=0.0 Nibs=5
                                + 1C3CE SB%GPH(00009) Type=0.0 Nibs=5
AVMMOV  =  18BCA SB%IO:         - 17CF3 SB%RD:(0012C) Type=1.1 Nibs=4 Dist=00ED7
AVS2DS  =  09708 TI%ERD         - 0FE77 SC%TRC(00430) Type=1.0 Nibs=4 Dist=0676F
AVS=C   =  01497 TI%UTL         - 06D66 SG%SYS(009C6) Type=1.1 Nibs=4 Dist=058CF
                                + 095CF TI%ERD(0023C) Type=0.1 Nibs=5
                                + 095DF TI%ERD(0024C) Type=0.1 Nibs=5
AVS=DO  =  01494 TI%UTL         - 026AF JP%PR1(0010F) Type=1.1 Nibs=4 Dist=0121B
                                + 0287B JP%PR1(002DB) Type=1.1 Nibs=4 Dist=013E7
                                + 03D8F JP%PR3(009B9) Type=1.1 Nibs=4 Dist=028FB
                                + 06764 SG%SYS(003C4) Type=1.1 Nibs=4 Dist=052D0
                                + 06C2B SG%SYS(0088B) Type=1.1 Nibs=4 Dist=05797
Adhead  =  0AA43 SB%FCN         - 0938F SG%EXC(00D47) Type=1.0 Nibs=4 Dist=016B4

B->TXT  =  1787E FH%TFM         -
B9605A  =  0AA85 TI%FX2         - 0F876 AB%ASN(0029C) Type=0.0 Nibs=5
B9605B  =  0AA9D TI%FX2         - 0BA78 AB%FCN(0043F) Type=0.1 Nibs=5
BABBAJ  =  0B866 AB%FCN         - 13552 PM%FLG(00109) Type=1.0 Nibs=4 Dist=07CEC
BACK    =  1BA4F MB%IMG         -
BACK1B  =  13B0C SC%DAT         -
BACK2B  =  13B0A SC%DAT         -
BACK3B  =  13B08 SC%DAT         -
BANG    =  08A48 SG%EXC         - 1EE7E SB%TAB(00947) Type=1.2 Nibs=5 Dist=16436
BASCHA  =  07741 JP%SYS         - 082F3 JP%EXC(00517) Type=1.1 Nibs=4 Dist=00BB2
BASCHK  =  0773E JP%SYS         -
BASE    =  0F953 AB%ASN         - 0AE1E AB%REG(00226) Type=1.1 Nibs=4 Dist=04B35
                                + 0BA16 AB%FCN(003DD) Type=1.1 Nibs=4 Dist=03F3D
BASICs  =  000B5 TI%EQU         - 074A1 JP%SYS(00572) Type=0.0 Nibs=2
BASKEY  =  06EC0 SG%SYS         - 0A2D6 SG%FXQ(00760) Type=1.1 Nibs=4 Dist=03416
                                + 0A5A7 SG%FXQ(00A31) Type=1.1 Nibs=4 Dist=036E7
BASTYP  =  1A3C6 SC%SUB         - 0FE54 SC%TRC(0040D) Type=0.1 Nibs=5
BBCOLL  =  01462 TI%UTL         - 17125 FH%TFM(00677) Type=0.1 Nibs=5
BEEP    =  0EA6E MN%BP:         - 1EDCA SB%TAB(00893) Type=1.2 Nibs=5 Dist=1035C
```

```
BEEPDC =  0584C SG%LDC        - 0EA64 MN%BP:(00073) Type=1.2 Nibs=5 Dist=09218
BEEPP  =  02AAB JP%PR1        - 0EA69 MN%BP:(00078) Type=1.2 Nibs=5 Dist=0BFBE
BF2BLD =  01894 SB%DSP        - 0248D MB%ROM(00009) Type=1.1 Nibs=4 Dist=00BF9
BF2DPP =  01BF6 SB%DSP        - 0037A SB%DVR(0037A) Type=1.1 Nibs=4 Dist=0187C
                              + 10130 JP%MEM(001D1) Type=0.1 Nibs=5
BF2DS+ =  01C08 SB%DSP        -
BF2DSP =  01C0E SB%DSP        - 002B5 SB%DVR(002B5) Type=1.1 Nibs=4 Dist=01959
                              + 00599 SB%DVR(00599) Type=1.1 Nibs=4 Dist=01675
                              + 01651 SB%CMD(0002A) Type=1.0 Nibs=3 Dist=005BD
                              + 0166F SB%CMD(00048) Type=1.1 Nibs=3 Dist=0059F
                              + 0649C SG%SYS(000FC) Type=1.1 Nibs=4 Dist=0488E
                              + 15238 MN%ED:(005B8) Type=0.1 Nibs=5
BF2ST+ =  18660 SB%IO:        - 0981A TI%ERD(00487) Type=0.1 Nibs=5
BF2STK =  18663 SB%IO:        - 06819 SG%SYS(00479) Type=0.1 Nibs=5
                              + 0A947 SB%FCN(0001C) Type=0.1 Nibs=5
BIASA+ =  0D52D SM%MTH        -
BIASA- =  0D52D SM%MTH        - 0CA24 JT%MTH(006FD) Type=1.1 Nibs=4 Dist=00B09
BIASC+ =  0D540 SM%MTH        - 0C9FC JT%MTH(006D5) Type=1.1 Nibs=4 Dist=00B44
BIASC- =  0D540 SM%MTH        - 0CA33 JT%MTH(0070C) Type=1.1 Nibs=4 Dist=00B0D
BIG    =  0B747 AB%FCN        - 0CAAD JT%MTH(00786) Type=1.0 Nibs=4 Dist=01366
BIG+   =  0B744 AB%FCN        - 1621F AB%CLC(00D00) Type=0.1 Nibs=5
BLANKC =  07818 JP%SYS        - 03F19 JP%PR3(00B43) Type=1.1 Nibs=4 Dist=038FF
                              + 03FBE JP%PR3(00BE8) Type=1.1 Nibs=4 Dist=0385A
                              + 1642C AB%BLD(00100) Type=0.1 Nibs=5
                              + 1804A SB%IO:(0028E) Type=0.1 Nibs=5
                              + 1CA3B MN%CD:(001C2) Type=0.1 Nibs=5
BLDB10 =  019BF SB%DSP        -
BLDBIT =  019BC SB%DSP        - 0011F SB%DVR(0011F) Type=0.1 Nibs=5
                              + 15168 MN%ED:(004E8) Type=0.1 Nibs=5
BLDCON =  16279 AB%CLC        - 15A8E AB%CLC(0056F) Type=1.1 Nibs=3 Dist=007EB
BLDDSP =  01898 SB%DSP        - 00126 SB%DVR(00126) Type=0.1 Nibs=5
                              + 01061 SB%KEY(0036A) Type=1.1 Nibs=4 Dist=00837
                              + 02580 MB%ROM(000FC) Type=1.0 Nibs=4 Dist=00CE8
                              + 072DC JP%SYS(003AD) Type=1.1 Nibs=4 Dist=05A44
                              + 09606 TI%ERD(00273) Type=1.1 Nibs=4 Dist=07D6E
                              + 1014A JP%MEM(001EB) Type=0.1 Nibs=5
                              + 14E13 MN%ED:(00193) Type=0.1 Nibs=5
                              + 14E3C MN%ED:(001BC) Type=0.1 Nibs=5
BLDLCD =  0189C SB%DSP        - 0012D SB%DVR(0012D) Type=0.1 Nibs=5
BLDNUM =  0F197 AB%EXP        - 1E5B4 SB%TAB(0007D) Type=1.2 Nibs=5 Dist=0F41D
                              + 1E5BD SB%TAB(00086) Type=1.2 Nibs=5 Dist=0F426
                              + 1E5C6 SB%TAB(0008F) Type=1.2 Nibs=5 Dist=0F42F
                              + 1E5CF SB%TAB(00098) Type=1.2 Nibs=5 Dist=0F438
                              + 1E5D8 SB%TAB(000A1) Type=1.2 Nibs=5 Dist=0F441
                              + 1E5E1 SB%TAB(000AA) Type=1.2 Nibs=5 Dist=0F44A
                              + 1E5EA SB%TAB(000B3) Type=1.2 Nibs=5 Dist=0F453
                              + 1E5F3 SB%TAB(000BC) Type=1.2 Nibs=5 Dist=0F45C
                              + 1E5FC SB%TAB(000C5) Type=1.2 Nibs=5 Dist=0F465
                              + 1E605 SB%TAB(000CE) Type=1.2 Nibs=5 Dist=0F46E
                              + 1E60E SB%TAB(000D7) Type=1.2 Nibs=5 Dist=0F477
                              + 1E644 SB%TAB(0010D) Type=1.2 Nibs=5 Dist=0F4AD
                              + 1E64D SB%TAB(00116) Type=1.2 Nibs=5 Dist=0F4B6
                              + 1E656 SB%TAB(0011F) Type=1.2 Nibs=5 Dist=0F4BF
                              + 1E65F SB%TAB(00128) Type=1.2 Nibs=5 Dist=0F4C8
                              + 1E668 SB%TAB(00131) Type=1.2 Nibs=5 Dist=0F4D1
                              + 1E671 SB%TAB(0013A) Type=1.2 Nibs=5 Dist=0F4DA
                              + 1E67A SB%TAB(00143) Type=1.2 Nibs=5 Dist=0F4E3
                              + 1E683 SB%TAB(0014C) Type=1.2 Nibs=5 Dist=0F4EC
                              + 1E68C SB%TAB(00155) Type=1.2 Nibs=5 Dist=0F4F5
                              + 1E695 SB%TAB(0015E) Type=1.2 Nibs=5 Dist=0F4FE
```

```
                                + 1E69E SB%TAB(00167) Type=1.2 Nibs=5 Dist=0F507
                                + 1E6A7 SB%TAB(00170) Type=1.2 Nibs=5 Dist=0F510
BLKOK  =  09E80 SG%FXQ          - 0354A JP%PR3(00174) Type=1.1 Nibs=4 Dist=06936
BLNKC+ =  07810 JP%SYS          - 06748 SG%SYS(003A8) Type=1.1 Nibs=4 Dist=010C8
                                + 0A478 SG%FXQ(00902) Type=1.1 Nibs=4 Dist=02C68
BLNKCK =  051C1 SG%LDC          - 0355D JP%PR3(00187) Type=1.1 Nibs=4 Dist=01C64
BOOLE  =  0B863 AB%FCN          -
BOPNM- =  1B864 MB%IMG          -
BOPWO5 =  1B87D MB%IMG          -
BP     =  0EADF MN%BP:          -
BP+    =  0EB3E MN%BP:          -
BP+C   =  0EB40 MN%BP:          -
BRT30  =  0DBE3 SM%MTH          -
BRTF   =  0DC15 SM%MTH          -
BSCEX2 =  0743A JP%SYS          -
BSCEXC =  07437 JP%SYS          - 028EB JP%PR1(0034B) Type=1.0 Nibs=4 Dist=04B4C
BSCEXT =  075CF JP%SYS          -
BSCXLP =  07451 JP%SYS          - 08A3A SG%EXC(003F2) Type=1.0 Nibs=4 Dist=015E9
BSERR  =  0939A TI%ERD          - 065E1 SG%SYS(00241) Type=1.0 Nibs=4 Dist=02DB9
                                + 06F86 JP%SYS(00057) Type=1.0 Nibs=4 Dist=02414
                                + 0822A JP%EXC(0044E) Type=1.0 Nibs=4 Dist=01170
                                + 0A09B SG%FXQ(00525) Type=1.0 Nibs=4 Dist=00D01
                                + 0CC22 JT%MTH(008FB) Type=1.0 Nibs=4 Dist=03888
                                + 142B2 SC%DAT(009FD) Type=0.1 Nibs=5
BSTYCK =  0FE43 SC%TRC          - 096E9 TI%ERD(00356) Type=1.1 Nibs=4 Dist=0675A
BUFASN =  118AB SC%FIL          -
BUFFIB =  118A2 SC%FIL          - 0A877 SG%FXQ(00D01) Type=1.1 Nibs=4 Dist=0702B
BUILD  =  16381 AB%BLD          - 158A7 AB%CLC(00388) Type=1.1 Nibs=4 Dist=00ADA
BYE    =  07FE2 JP%EXC          - 1DDC0 JP%TAB(000BC) Type=1.2 Nibs=5 Dist=15DDE
BYTscA =  1BA4A MB%IMG          - 1BAF6 MB%USG(00052) Type=1.1 Nibs=3 Dist=000AC
                                + 1BE5B MB%USG(003B7) Type=1.1 Nibs=3 Dist=00411
Bf2Dsp =  01650 SB%CMD          - 0104D SB%KEY(00356) Type=1.1 Nibs=3 Dist=00603
BitsOK =  00001 SB%DSP          -
BldIM+ =  1BA6A MB%IMG          - 1BACD MB%USG(00029) Type=1.1 Nibs=3 Dist=00063
                                + 1C023 MB%USG(0057F) Type=1.0 Nibs=3 Dist=005B9
BldIMA =  1BA66 MB%IMG          -
BldIMG =  1BA68 MB%IMG          -
BsErr  =  142B0 SC%DAT          - 12243 SC%FIL(011F0) Type=1.0 Nibs=4 Dist=0206D
                                + 16B16 FH%TFM(00068) Type=1.0 Nibs=4 Dist=02866
                                + 19708 SC%SUB(00996) Type=1.0 Nibs=4 Dist=05458
C+A2D1 =  1C053 MB%USG          - 1B842 MB%IMG(003FC) Type=1.1 Nibs=4 Dist=00811
C=MAIN =  0A010 SG%FXQ          - 06475 SG%SYS(000D5) Type=1.1 Nibs=4 Dist=03B9B
                                + 066C1 SG%SYS(00321) Type=1.1 Nibs=4 Dist=0394F
                                + 10C2C MN%CNF(00A9A) Type=1.1 Nibs=4 Dist=06C1C
                                + 10DFA MN%CNF(00C68) Type=1.1 Nibs=4 Dist=06DEA
C=RAME =  10489 MN%CNF          - 00257 SB%DVR(00257) Type=0.1 Nibs=5
                                + 07D2B JP%SYS(00DFC) Type=0.1 Nibs=5
                                + 07DC7 JP%SYS(00E98) Type=0.1 Nibs=5
                                + 1C6E6 SG%POK(001F8) Type=0.1 Nibs=5
C=SCRL =  02266 SB%DSP          - 00367 SB%DVR(00367) Type=1.1 Nibs=4 Dist=01EFF
CALBIN =  18D8C SC%SUB          -
CALC   =  1588E AB%CLC          - 004C3 SB%DVR(004C3) Type=0.1 Nibs=5
CALCj  =  004C1 SB%DVR          - 010C1 SB%KEY(003CA) Type=1.0 Nibs=4 Dist=00C00
CALFX1 =  1DBD2 TI%FX4          - 192AF SC%SUB(0053D) Type=1.1 Nibs=4 Dist=04923
CALL   =  18DAE SC%SUB          - 1EE63 SB%TAB(0092C) Type=1.2 Nibs=5 Dist=060B5
CALLDC =  058BC SG%LDC          - 18DA4 SC%SUB(00032) Type=1.2 Nibs=5 Dist=134E8
CALLP  =  0389C JP%PR3          - 18DA9 SC%SUB(00037) Type=1.2 Nibs=5 Dist=1550D
CALSTK =  2F5AD SB%RAM          - 0A707 SG%FXQ(00B91) Type=0.0 Nibs=5
                                + 0B296 AB%REG(0069E) Type=0.0 Nibs=5
```

```
                                    + 1903E SC%SUB(002CC) Type=0.0 Nibs=5
                                    + 194C1 SC%SUB(0074F) Type=0.0 Nibs=5
                                    + 19797 SC%SUB(00A25) Type=0.0 Nibs=5
CARYSV =  170CB FH%TFM              -
CAT    =  0642C SG%SYS              - 1EDEE SB%TAB(008B7) Type=1.2 Nibs=5 Dist=189C2
CAT$   =  06668 SG%SYS              - 1DDC9 JP%TAB(000C5) Type=1.2 Nibs=5 Dist=17761
CAT$20 =  06746 SG%SYS              - 1D8DE MN%CD:(01065) Type=0.1 Nibs=5
CAT$65 =  067FE SG%SYS              -
CAT$66 =  0680A SG%SYS              - 1D90E MN%CD:(01095) Type=0.1 Nibs=5
CAT$83 =  068D4 SG%SYS              - 0A4AF SG%FXQ(00939) Type=1.1 Nibs=4 Dist=03BDB
CAT$90 =  068D7 SG%SYS              - 0A4C2 SG%FXQ(0094C) Type=1.1 Nibs=4 Dist=03BEB
CAT100 =  06658 SG%SYS              - 1D8F1 MN%CD:(01078) Type=0.1 Nibs=5
CATC++ =  03F66 JP%PR3              - 09CE3 SG%FXQ(0016D) Type=1.1 Nibs=4 Dist=05D7D
CATCH+ =  03F69 JP%PR3              -
CATCHR =  03F70 JP%PR3              -
CATCRD =  1D7C8 MN%CD:              - 0641D SG%SYS(0007D) Type=0.1 Nibs=5
CATEDT =  06435 SG%SYS              - 0A5DF SG%FXQ(00A69) Type=1.0 Nibs=4 Dist=041AA
CATP   =  03A8C JP%PR3              - 06427 SG%SYS(00087) Type=1.2 Nibs=5 Dist=0299B
CEIL   =  0E87B PM%STA              - 1E9A4 SB%TAB(0046D) Type=1.2 Nibs=5 Dist=10129
CFLAG  =  13492 PM%FLG              - 1EE6C SB%TAB(00935) Type=1.2 Nibs=5 Dist=0B9DA
CHAIN  =  06F39 JP%SYS              - 1DDFF JP%TAB(000FB) Type=1.2 Nibs=5 Dist=16EC6
CHAIN* =  07C31 JP%SYS              - 1A331 SC%SUB(015BF) Type=0.1 Nibs=5
CHAIN+ =  07C12 JP%SYS              -
CHAIN- =  07C1C JP%SYS              - 082FE JP%EXC(00522) Type=1.1 Nibs=3 Dist=006E2
                                    + 1784A FH%TFM(00D9C) Type=0.1 Nibs=5
CHAINP =  03DAE JP%PR3              - 06F34 JP%SYS(00005) Type=1.2 Nibs=5 Dist=03186
CHARER =  1A535 SB%EXC              - 1A880 SC%REN(00304) Type=1.0 Nibs=3 Dist=0034B
CHARST =  1A4CE SB%EXC              - 1DDF6 JP%TAB(000F2) Type=1.2 Nibs=5 Dist=03928
CHEDIT =  14C99 MN%ED:              - 00134 SB%DVR(00134) Type=0.1 Nibs=5
                                    + 16A66 AB%ED:(003F4) Type=1.1 Nibs=4 Dist=01DCD
                                    + 1871A SB%ID:(0095E) Type=1.1 Nibs=4 Dist=03A81
                                    + 18CE0 MN%LCK(00088) Type=1.1 Nibs=4 Dist=04047
CHIRP  =  0EC5A MN%BP:              - 095EF TI%ERD(0025C) Type=1.1 Nibs=4 Dist=0566B
                                    + 14E0D MN%ED:(0018D) Type=1.0 Nibs=4 Dist=061B3
CHKDON =  13E32 SC%DAT              - 17D87 SB%RD:(001C0) Type=1.1 Nibs=4 Dist=03F55
CHKEOL =  13D6D SC%DAT              - 11140 SC%FIL(000ED) Type=1.1 Nibs=4 Dist=02C2D
                                    + 11152 SC%FIL(000FF) Type=1.1 Nibs=4 Dist=02C1B
CHKPSF =  0777A JP%SYS              - 06964 SG%SYS(005C4) Type=1.1 Nibs=4 Dist=00E16
                                    + 069B0 SG%SYS(00610) Type=1.1 Nibs=4 Dist=00DCA
                                    + 06D34 SG%SYS(00994) Type=1.1 Nibs=4 Dist=00A46
                                    + 0FF64 JP%MEM(00005) Type=0.1 Nibs=5
                                    + 1A58E SC%REN(00012) Type=0.1 Nibs=5
CHKRSP =  12236 SC%FIL              - 13AF4 SC%DAT(0023F) Type=1.0 Nibs=4 Dist=018BE
CHKSPC =  012AE TI%UTL              - 017DD SB%CMD(001B6) Type=1.1 Nibs=3 Dist=0052F
                                    + 0F946 AB%ASN(0036C) Type=0.1 Nibs=5
                                    + 18FF2 SC%SUB(00280) Type=0.1 Nibs=5
                                    + 19256 SC%SUB(004E4) Type=0.1 Nibs=5
CHKSPF =  012C2 TI%UTL              -
CHKSTK =  13DB6 SC%DAT              - 18DEA SC%SUB(00078) Type=1.1 Nibs=4 Dist=05034
                                    + 1A191 SC%SUB(0141F) Type=1.1 Nibs=4 Dist=063DB
CHKTYP =  1A394 SC%SUB              -
CHKmem =  012C7 TI%UTL              - 0CC41 JT%MTH(0091A) Type=0.1 Nibs=5
CHNNSV =  2F96F SB%RAM              - 1143A SC%FIL(003E7) Type=0.0 Nibs=5
                                    + 12065 SC%FIL(01012) Type=0.0 Nibs=5
                                    + 13E13 SC%DAT(0055E) Type=0.0 Nibs=5
                                    + 19CDB SC%SUB(00F69) Type=0.0 Nibs=5
                                    + 19F6C SC%SUB(011FA) Type=0.0 Nibs=5
CHNHED =  0F579 AB%EXP              - 0B2BD AB%REG(006C5) Type=1.1 Nibs=4 Dist=042BC
CHNLST =  2F5BE SB%RAM              - 0B25A AB%REG(00662) Type=0.0 Nibs=5 Offset=    175
                                    + 0F593 AB%EXP(0041B) Type=0.0 Nibs=5 Offset=   -455
```

```
                                + 1917E SC%SUB(0040C) Type=0.0 Nibs=5
                                + 197B0 SC%SUB(00A3E) Type=0.0 Nibs=5
CHR$    =  0BE26 AB%FCN         - 1EB66 SB%TAB(0062F) Type=1.2 Nibs=5 Dist=12D40
CHRBUF  =  0199F SB%DSP         - 0AA16 SB%FCN(000EB) Type=0.1 Nibs=5
CHRST$  =  0A9F8 SB%FCN         - 1DE08 JP%TAB(00104) Type=1.2 Nibs=5 Dist=13410
CK"ON"  =  076AD JP%SYS         - 004DC SB%DVR(004DC) Type=1.1 Nibs=4 Dist=071D1
                                + 06B09 SG%SYS(00769) Type=1.1 Nibs=4 Dist=00BA4
                                + 17ECE SB%IO:(00112) Type=0.1 Nibs=5
                                + 18708 SB%IO:(0094C) Type=0.1 Nibs=5
                                + 1AD14 SG%KEY(00273) Type=0.1 Nibs=5
                                + 1C150 MB%USG(006AC) Type=0.1 Nibs=5
CKINF-  =  18534 SB%IO:         - 0966B TI%ERD(002D8) Type=0.1 Nibs=5
CKINF0  =  18542 SB%IO:         - 06B1E SG%SYS(0077E) Type=0.1 Nibs=5
                                + 1B4FD MB%IMG(000B7) Type=1.1 Nibs=4 Dist=02FBB
                                + 1C093 MB%USG(005EF) Type=1.1 Nibs=4 Dist=03B51
                                + 1C2E0 MB%USG(0083C) Type=1.1 Nibs=4 Dist=03D9E
CKSREQ  =  00721 SB%DVR         - 01DAB SB%DSP(0051B) Type=1.0 Nibs=4 Dist=0168A
                                + 0751B JP%SYS(005EC) Type=1.1 Nibs=4 Dist=06DFA
                                + 0ED98 MN%UTL(0012A) Type=0.1 Nibs=5
                                + 150E9 MN%ED:(00469) Type=0.1 Nibs=5
CKSUM1  =  0000D SB%DVR         -
CKSUM2  =  0AA81 Define         -
CKSUM3  =  153A9 Define         -
CKSUM4  =  1DBA6 Define         -
CKTMOU  =  0EF86 MN%UTL         - 00737 SB%DVR(00737) Type=0.1 Nibs=5
CL-IDS  =  0F0D8 MN%UTL         - 10EBA MN%CNF(00D28) Type=1.1 Nibs=4 Dist=01DE2
CLOST   =  1B9CD MB%IMG         - 1C1E5 MB%USG(00741) Type=1.1 Nibs=4 Dist=00818
CLOST+  =  1B9CA MB%IMG         - 1BAC3 MB%USG(0001F) Type=1.1 Nibs=3 Dist=000F9
                                + 1C045 MB%USG(005A1) Type=1.1 Nibs=3 Dist=0067B
CLAIMp  =  0338C JP%PR2         - 17B2E MN%FRP(000FB) Type=1.2 Nibs=5 Dist=147A2
CLASS   =  0B721 AB%FCN         - 1DE1A JP%TAB(00116) Type=1.2 Nibs=5 Dist=126F9
CLASSA  =  0D590 SM%MTH         - 0B72A AB%FCN(000F1) Type=1.1 Nibs=4 Dist=01E66
CLCBFR  =  2F576 SB%RAM         - 01819 SB%CMD(001F2) Type=0.0 Nibs=4
                                + 1552C AB%CLC(0000D) Type=0.0 Nibs=4
                                + 16852 AB%ED:(001E0) Type=0.0 Nibs=5
                                + 16994 AB%ED:(00322) Type=0.0 Nibs=5
                                + 16A0C AB%ED:(0039A) Type=0.0 Nibs=5
CLCBTS  =  16308 AB%CLC         - 16811 AB%ED:(0019F) Type=1.1 Nibs=3 Dist=00509
CLCCFR  =  16717 AB%ED:         - 15937 AB%CLC(00418) Type=1.0 Nibs=4 Dist=00DE0
CLCERR  =  158EE AB%CLC         - 00343 SB%DVR(00343) Type=0.1 Nibs=5
CLCET1  =  16672 AB%ED:         - 1577F AB%CLC(00260) Type=0.0 Nibs=5
CLCET2  =  16A7A AB%ED:         - 16A6F AB%ED:(003FD) Type=0.0 Nibs=5 Offset=    -52
CLCEXP  =  162CC AB%CLC         - 1680D AB%ED:(0019B) Type=1.1 Nibs=3 Dist=00541
CLCOLL  =  01448 TI%UTL         - 0038E SB%DVR(0038E) Type=1.1 Nibs=4 Dist=010BA
                                + 0947E TI%ERD(000EB) Type=0.1 Nibs=5
CLCSTK  =  2F585 SB%RAM         - 0144A TI%UTL(00319) Type=0.0 Nibs=5
                                + 15DAF AB%CLC(00890) Type=0.0 Nibs=5
                                + 15F4A AB%CLC(00A2B) Type=0.0 Nibs=5
                                + 1640D AB%BLD(000E1) Type=0.0 Nibs=2
                                + 165C2 AB%BLD(00296) Type=0.0 Nibs=5
                                + 16830 AB%ED:(001BE) Type=0.0 Nibs=5
                                + 16918 AB%ED:(002A6) Type=0.0 Nibs=5
CLCSTR  =  1820E SB%IO:         - 165E2 AB%BLD(002B6) Type=1.1 Nibs=4 Dist=01C2C
CLDST1  =  001EE SB%DVR         - 00004 SB%DVR(00004) Type=0.0 Nibs=5
                                + 0109C SB%KEY(003A5) Type=1.0 Nibs=4 Dist=00EAE
CLKPRS  =  037A5 JP%PR3         - 12C95 MN%TM:(0074B) Type=1.2 Nibs=5 Dist=0F4F0
                                + 12D03 MN%TM:(007B9) Type=1.2 Nibs=5 Dist=0F55E
                                + 12D27 MN%TM:(007DD) Type=1.2 Nibs=5 Dist=0F582
                                + 12D97 MN%TM:(0084D) Type=1.2 Nibs=5 Dist=0F5F2
CLKSPD  =  0E9F1 MN%BP:         - 10D63 MN%CNF(00BD1) Type=1.1 Nibs=4 Dist=02372
```

```
CLLINK =  1A72C SC%REN        - 06985 SG%SYS(005E5) Type=0.1 Nibs=5
                              + 0FF72 JP%MEM(00013) Type=0.1 Nibs=5
CLMPRO =  06EFF SG%SYS        - 17B6E MN%FRP(0013B) Type=0.1 Nibs=5
CLOSE# =  12071 SC%FIL        -
CLOSEA =  120E4 SC%FIL        - 0766A JP%SYS(0073B) Type=0.1 Nibs=5
                              + 10E70 MN%CNF(00CDE) Type=1.1 Nibs=4 Dist=01274
                              + 1961C SC%SUB(008AA) Type=1.1 Nibs=4 Dist=07538
CLOSEF =  12087 SC%FIL        - 16F8E FH%TFM(004E0) Type=1.1 Nibs=4 Dist=04F07
CLPSTK =  07D29 JP%SYS        - 0697F SG%SYS(005DF) Type=1.1 Nibs=4 Dist=013AA
                              + 070A5 JP%SYS(00176) Type=1.1 Nibs=4 Dist=00C84
                              + 0A4ED SG%FXQ(00977) Type=1.1 Nibs=4 Dist=027C4
                              + 0FF6B JP%MEM(0000C) Type=0.1 Nibs=5
                              + 19615 SC%SUB(008A3) Type=0.1 Nibs=5
CLRFRC =  0C6F4 JT%MTH        - 09ED2 SG%FXQ(0035C) Type=1.1 Nibs=4 Dist=02822
                              + 09EE5 SG%FXQ(0036F) Type=1.1 Nibs=4 Dist=0280F
                              + 0E8A2 PM%STA(00A73) Type=1.0 Nibs=4 Dist=021AE
                              + 12BBA MN%TM:(00670) Type=1.1 Nibs=4 Dist=064C6
CLRPRM =  04827 SB%EXP        - 038ED JP%PR3(00517) Type=1.1 Nibs=4 Dist=00F3A
CLRSTK =  07D3B JP%SYS        - 07665 JP%SYS(00736) Type=1.1 Nibs=3 Dist=006D6
CLRXDS =  01024 SB%KEY        - 10D50 MN%CNF(00BBE) Type=0.1 Nibs=5
                              + 1555D AB%CLC(0003E) Type=0.1 Nibs=5
                              + 1670C AB%ED:(0009A) Type=0.1 Nibs=5
CLSTAT =  0DE4E PM%STA        - 1DE2C JP%TAB(00128) Type=1.2 Nibs=5 Dist=0FFDE
CLSUSP =  07991 JP%SYS        - 0911B SG%EXC(00AD3) Type=1.1 Nibs=4 Dist=0178A
CMD1ST =  01654 SB%CMD        - 18701 SB%IO:(00945) Type=0.1 Nibs=5
                              + 18A71 SB%IO:(00CB5) Type=0.1 Nibs=5
CMDFND =  01693 SB%CMD        - 18A7C SB%IO:(00CC0) Type=0.1 Nibs=5
CMDINI =  016D1 SB%CMD        - 18A6A SB%IO:(00CAE) Type=0.1 Nibs=5
CMDPR" =  01627 SB%CMD        - 01669 SB%CMD(00042) Type=0.0 Nibs=5
CMDPTR =  2F6D4 SB%RAM        - 01658 SB%CMD(00031) Type=0.0 Nibs=5
                              + 01695 SB%CMD(0006E) Type=0.0 Nibs=5
                              + 016DE SB%CMD(000B7) Type=0.0 Nibs=4
CMDS10 =  01667 SB%CMD        -
CMDS20 =  01672 SB%CMD        -
CMDST? =  016C4 SB%CMD        -
CMDSTK =  0163B SB%CMD        - 004A1 SB%DVR(004A1) Type=1.1 Nibs=4 Dist=0119A
CMOSTV =  0168F SB%RAM        - 0009B SB%DVR(0009B) Type=0.0 Nibs=4
                              + 001F7 SB%DVR(001F7) Type=0.0 Nibs=4
CMOSTW =  2F438 SB%RAM        - 00093 SB%DVR(00093) Type=0.0 Nibs=2
                              + 001F0 SB%DVR(001F0) Type=0.0 Nibs=5
CMPB#C =  045BE SB%EXP        - 06061 SB%EXD(0073F) Type=1.1 Nibs=4 Dist=01AA3
CMPLX  =  0BC1C AB%FCN        - 04401 SB%EXP(0042C) Type=0.0 Nibs=5 Offset=     -2
                              + 1E9EC SB%TAB(004B5) Type=1.2 Nibs=5 Dist=12DD0
CMPT   =  125B2 MN%TM:        -
CMPTIM =  1260D MN%TM:        - 1C8FA MN%CD:(00081) Type=0.1 Nibs=5
CNFFND =  109AC MN%CNF        - 17A47 MN%FRP(00014) Type=1.1 Nibs=4 Dist=0709B
                              + 17B42 MN%FRP(0010F) Type=1.1 Nibs=4 Dist=07196
                              + 1C6F7 SG%POK(00209) Type=0.1 Nibs=5
CNFLCT =  0BD15 AB%FCN        - 0AD67 AB%REG(0016F) Type=1.0 Nibs=4 Dist=00FAE
                              + 0E908 PM%STA(00AD9) Type=1.0 Nibs=4 Dist=02BF3
                              + 13558 PM%FLG(0010F) Type=1.0 Nibs=4 Dist=07843
CNTADR =  2F67E SB%RAM        - 07200 JP%SYS(002D1) Type=0.0 Nibs=5
                              + 07362 JP%SYS(00433) Type=0.0 Nibs=5
                              + 07D5A JP%SYS(00E2B) Type=0.0 Nibs=5
                              + 090CD SG%EXC(00A85) Type=0.0 Nibs=5
CNTCHR =  15876 AB%CLC        - 16933 AB%ED:(002C1) Type=1.1 Nibs=4 Dist=010BD
CNTCK2 =  071FE JP%SYS        - 09121 SG%EXC(00AD9) Type=1.1 Nibs=4 Dist=01F23
CNTCUR =  07388 JP%SYS        -
CNTRST =  0F073 MN%UTL        - 1DE35 JP%TAB(00131) Type=1.2 Nibs=5 Dist=0EDC2
CNV2UC =  03FB4 JP%PR3        - 02BB2 JP%PR1(00612) Type=1.1 Nibs=4 Dist=01402
```

```
CNVUCR =  152A7 MN%ED:          - 0AA6A SB%FCN(0013F) Type=0.1 Nibs=5
CNVWUC =  03FB8 JP%PR3          - 09C28 SG%FXQ(000B2) Type=1.1 Nibs=4 Dist=05C70
COLDSM =  0068F SB%DVR          - 002AE SB%DVR(002AE) Type=0.0 Nibs=5
COLDST =  00000 SB%DVR          - 1072E MN%CNF(0059C) Type=0.1 Nibs=5
COLLAP =  091FB SG%EXC          - 00387 SB%DVR(00387) Type=0.1 Nibs=5
                                + 0719A JP%SYS(0026B) Type=1.1 Nibs=4 Dist=02061
                                + 09479 TI%ERD(000E6) Type=1.1 Nibs=3 Dist=0027E
                                + 0AC40 AB%REG(00048) Type=1.1 Nibs=4 Dist=01A45
                                + 0E233 PM%STA(00404) Type=1.1 Nibs=4 Dist=05038
                                + 0F17A AB%EXP(00002) Type=1.1 Nibs=4 Dist=05F7F
                                + 17F4F SB%IO:(00193) Type=0.1 Nibs=5
COMCK  =  036CD JP%PR3          - 030FA JP%PR2(004D4) Type=1.1 Nibs=3 Dist=005D3
COMCK+ =  032AE JP%PR2          - 02B6F JP%PR1(005CF) Type=1.1 Nibs=3 Dist=0073F
                                + 033E5 JP%PR3(0000F) Type=1.1 Nibs=3 Dist=00137
                                + 03609 JP%PR3(00233) Type=1.1 Nibs=3 Dist=0035B
                                + 03760 JP%PR3(0038A) Type=1.1 Nibs=3 Dist=004B2
                                + 03793 JP%PR3(003BD) Type=1.1 Nibs=3 Dist=004E5
                                + 039D6 JP%PR3(00600) Type=1.1 Nibs=3 Dist=00728
                                + 039E6 JP%PR3(00610) Type=1.1 Nibs=3 Dist=00738
                                + 03A65 JP%PR3(0068F) Type=1.1 Nibs=3 Dist=007B7
COMCK1 =  036D1 JP%PR3          - 02ACE JP%PR1(0052E) Type=1.1 Nibs=4 Dist=00C03
                                + 0324D JP%PR2(00627) Type=1.1 Nibs=3 Dist=00484
                                + 032AF JP%PR2(00689) Type=1.1 Nibs=3 Dist=00422
                                + 0526A SG%LDC(0030C) Type=1.0 Nibs=4 Dist=01B99
COMCK0 =  032AA JP%PR2          - 02BA6 JP%PR1(00606) Type=1.1 Nibs=3 Dist=00704
                                + 03815 JP%PR3(0043F) Type=1.1 Nibs=3 Dist=0056B
                                + 03DD6 JP%PR3(00A00) Type=1.1 Nibs=4 Dist=00B2C
COMPAR =  0B7C5 AB%FCN          - 1EA7C SB%TAB(00545) Type=1.2 Nibs=5 Dist=132B7
COMPIL =  160FA AB%CLC          - 167EE AB%ED:(0017C) Type=1.1 Nibs=3 Dist=006F4
COMPL# =  07870 JP%SYS          - 0FEFD SC%TRC(004B6) Type=0.1 Nibs=5
                                + 19DC8 SC%SUB(01056) Type=0.1 Nibs=5
CONCAT =  0B6CF AB%FCN          - 1EA73 SB%TAB(0053C) Type=1.2 Nibs=5 Dist=133A4
CONCOM =  0467E SB%EXP          - 031D9 JP%PR2(005B3) Type=1.1 Nibs=4 Dist=014A5
CONF   =  10212 MN%CNF          - 0023E SB%DVR(0023E) Type=0.1 Nibs=5
                                + 010D1 SB%KEY(003DA) Type=0.1 Nibs=5
CONFMN =  010C5 SB%KEY          - 16883 AB%ED:(00211) Type=0.1 Nibs=5
CONFS3 =  10215 MN%CNF          - 17B1F MN%FRP(000EC) Type=1.1 Nibs=4 Dist=0790A
CONFST =  2F9E6 SB%RAM          - 10743 MN%CNF(005B1) Type=0.0 Nibs=4
                                + 107D9 MN%CNF(00647) Type=0.0 Nibs=5
                                + 10880 MN%CNF(006EE) Type=0.0 Nibs=4 Offset=      7
                                + 109AE MN%CNF(0081C) Type=0.0 Nibs=5
                                + 10AEC MN%CNF(0095A) Type=0.0 Nibs=5 Offset=      2
                                + 10C23 MN%CNF(00A91) Type=0.0 Nibs=5 Offset=     17
                                + 10CF1 MN%CNF(00B5F) Type=0.0 Nibs=5
CONT   =  06F5A JP%SYS          - 1DE3E JP%TAB(0013A) Type=1.2 Nibs=5 Dist=16EE4
CONTK  =  06F40 JP%SYS          - 004B7 SB%DVR(004B7) Type=1.0 Nibs=4 Dist=06A89
CONTP  =  03C2C JP%PR3          - 06F55 JP%SYS(00026) Type=1.2 Nibs=5 Dist=03329
CONVUC =  152AA MN%ED:          - 03F6B JP%PR3(00B95) Type=0.1 Nibs=5
                                + 08DF7 SG%EXC(007AF) Type=0.1 Nibs=5
                                + 1BA93 MB%IMG(0064D) Type=1.1 Nibs=4 Dist=067E9
                                + 1C5FC SG%POK(0010E) Type=1.1 Nibs=4 Dist=07352
COPY   =  081CB JP%EXC          - 1EBFF SB%TAB(006C8) Type=1.2 Nibs=5 Dist=16A34
COPYDC =  05722 SG%LDC          - 081C1 JP%EXC(003E5) Type=1.2 Nibs=5 Dist=02A9F
COPYP  =  03B0C JP%PR3          - 081C6 JP%EXC(003EA) Type=1.2 Nibs=5 Dist=046BA
COPYu  =  08269 JP%EXC          - 07038 JP%SYS(00109) Type=1.1 Nibs=4 Dist=01231
                                + 16CC6 FH%TFM(00218) Type=0.1 Nibs=5
CORR   =  0E35B PM%STA          - 1DE47 JP%TAB(00143) Type=1.2 Nibs=5 Dist=0FAEC
CORUPT =  09083 SG%EXC          - 04A24 AB%LEX(0013E) Type=1.0 Nibs=4 Dist=0465F
                                + 17B29 MN%FRP(000F6) Type=0.1 Nibs=5
                                + 1AEFC SB%VAL(0018E) Type=0.1 Nibs=5
```

```
COS     =  0BE67 AB%FCN        - 1EAF1 SB%TAB(005BA) Type=1.2 Nibs=5 Dist=12C8A
COS12   =  0D721 SM%MTH        - 0BE70 AB%FCN(00837) Type=1.1 Nibs=4 Dist=018B1
COS15   =  0D725 SM%MTH        -
COUNTC  =  1C346 MB%USG        -
CPL#10  =  07887 JP%SYS        - 07B59 JP%SYS(00C2A) Type=1.1 Nibs=3 Dist=002D2
                               + 09130 SG%EXC(00AE8) Type=1.1 Nibs=4 Dist=018A9
                               + 096F6 TI%ERD(00363) Type=1.1 Nibs=4 Dist=01E6F
                               + 0FE91 SC%TRC(0044A) Type=0.1 Nibs=5
                               + 0FEA1 SC%TRC(0045A) Type=0.1 Nibs=5
                               + 1A665 SC%REN(000E9) Type=0.1 Nibs=5
CPL#15  =  0788A JP%SYS        - 0729C JP%SYS(00360) Type=1.1 Nibs=3 Dist=005EE
CPOLL   =  0BC36 AB%FCN        - 0F68E AB%ASN(000B4) Type=1.1 Nibs=4 Dist=03A58
CPOLLF  =  0BC20 AB%FCN        -
CPOLLR  =  0BC27 AB%FCN        -
CPYERR  =  08407 JP%EXC        -
CR      =  2C000 TI%EQU        - 1CEAB MN%CD:(00632) Type=0.0 Nibs=5 Offset=     20
                               + 1D093 MN%CD:(0081A) Type=0.0 Nibs=5 Offset=     20
                               + 1D09E MN%CD:(00825) Type=0.0 Nibs=2
                               + 1D146 MN%CD:(008CD) Type=0.0 Nibs=5 Offset=     16
                               + 1D171 MN%CD:(008F8) Type=0.0 Nibs=5 Offset=     16
                               + 1D1D6 MN%CD:(0095D) Type=0.0 Nibs=5 Offset=     20
CR-ARR  =  0B43E AB%REG        - 0F8E5 AB%ASN(0030B) Type=1.1 Nibs=4 Dist=044A7
                               + 19408 SC%SUB(00696) Type=0.1 Nibs=5
CR-VAR  =  0B2BB AB%REG        - 0F841 AB%ASN(00267) Type=1.1 Nibs=4 Dist=04586
                               + 0F8D8 AB%ASN(002FE) Type=1.1 Nibs=4 Dist=0461D
                               + 192E8 SC%SUB(00576) Type=0.1 Nibs=5
                               + 193D8 SC%SUB(00666) Type=0.1 Nibs=5
CRDFIL  =  1D21D MN%CD:        - 08433 JP%EXC(00657) Type=0.1 Nibs=5
CREADC  =  0564E SG%LDC        - 1159D SC%FIL(0054A) Type=1.2 Nibs=5 Dist=0BF4F
CREATE  =  115A7 SC%FIL        - 1DE59 JP%TAB(00155) Type=1.2 Nibs=5 Dist=0C8B2
CREATF  =  084C1 JP%EXC        - 08BCC SG%EXC(00584) Type=1.1 Nibs=3 Dist=0070B
                               + 11FFA SC%FIL(00FA7) Type=0.1 Nibs=5
                               + 1D23F MN%CD:(009C6) Type=0.1 Nibs=5
CREATP  =  03A4E JP%PR3        - 115A2 SC%FIL(0054F) Type=1.2 Nibs=5 Dist=0DB54
CRETF+  =  084C4 JP%EXC        - 0A56A SG%FXQ(009F4) Type=1.1 Nibs=4 Dist=020A6
                               + 1178A SC%FIL(00737) Type=0.1 Nibs=5
CRETM5  =  084E1 JP%EXC        - 0A237 SG%FXQ(006C1) Type=1.1 Nibs=4 Dist=01D56
CRFSB-  =  11664 SC%FIL        -
CRGJMP  =  0307A JP%PR2        - 027DD JP%PR1(0023D) Type=1.1 Nibs=4 Dist=0089D
                               + 12407 JP%POL(000EC) Type=0.1 Nibs=5
CRLFND  =  0229E SB%DSP        - 010B2 SB%KEY(003BB) Type=1.1 Nibs=4 Dist=011EC
                               + 06660 SG%SYS(002C0) Type=1.0 Nibs=4 Dist=043C2
                               + 094BB TI%ERD(00128) Type=1.1 Nibs=4 Dist=0721D
                               + 095D6 TI%ERD(00243) Type=1.1 Nibs=4 Dist=07338
                               + 1018D JP%MEM(0022E) Type=0.1 Nibs=5
                               + 1CED0 MN%CD:(00657) Type=0.1 Nibs=5
CRLFOF  =  02296 SB%DSP        - 02631 JP%PR1(00091) Type=1.1 Nibs=3 Dist=0039B
                               + 0731F JP%SYS(003F0) Type=1.0 Nibs=4 Dist=05089
                               + 15556 AB%CLC(00037) Type=0.1 Nibs=5
                               + 16359 AB%BLD(0002D) Type=0.1 Nibs=5
                               + 18A61 SB%IO:(00CA5) Type=0.1 Nibs=5
CRLFPR  =  00328 SB%DVR        - 0164B SB%CMD(00024) Type=0.0 Nibs=5
CRLFSD  =  022A2 SB%DSP        - 08C7B SG%EXC(00633) Type=1.0 Nibs=4 Dist=069D9
                               + 09600 TI%ERD(0026D) Type=1.1 Nibs=4 Dist=0735E
                               + 0FE04 SC%TRC(003BD) Type=0.1 Nibs=5
CRTF    =  116C1 SC%FIL        - 16DEE FH%TFM(00340) Type=1.1 Nibs=4 Dist=0572D
CRTF50  =  1175D SC%FIL        -
CSETUP  =  0BBA3 AB%FCN        - 0F2EA AB%EXP(00172) Type=1.0 Nibs=4 Dist=03747
CSL2RO  =  1BA16 MB%IMG        - 1BF5E MB%USG(004BA) Type=1.1 Nibs=3 Dist=00548
                               + 1C072 MB%USG(005CE) Type=1.1 Nibs=3 Dist=0065C
```

```
                                    + 1C38A MB%USG(008E6) Type=1.1 Nibs=4 Dist=00974
CSL9R0 =  1BA0D MB%IMG              -
CSLC1  =  1B441 AB%UTL              -
CSLC10 =  1B418 AB%UTL              -
CSLC11 =  1B41B AB%UTL              -
CSLC12 =  1B41E AB%UTL              -
CSLC13 =  1B421 AB%UTL              -
CSLC14 =  1B424 AB%UTL              -
CSLC15 =  1B427 AB%UTL              -
CSLC2  =  1B43E AB%UTL              -
CSLC3  =  1B43B AB%UTL              - 10601 MN%CNF(0046F) Type=0.1 Nibs=5
CSLC4  =  1B438 AB%UTL              -
CSLC5  =  1B435 AB%UTL              - 0A926 SG%FXQ(00DB0) Type=0.1 Nibs=5
                                    + 172B4 FH%TFM(00806) Type=1.1 Nibs=4 Dist=04181
CSLC6  =  1B432 AB%UTL              -
CSLC7  =  1B42F AB%UTL              -
CSLC8  =  1B42C AB%UTL              -
CSLC9  =  1B415 AB%UTL              -
CSLW3  =  0ED43 MN%UTL              - 14FBB MN%ED:(0033B) Type=1.1 Nibs=4 Dist=06278
CSLW4  =  0ED40 MN%UTL              - 1028A MN%CNF(000F8) Type=1.1 Nibs=4 Dist=0154A
CSLW5  =  0ED3D MN%UTL              - 01195 TI%UTL(00064) Type=0.1 Nibs=5
                                    + 0EB46 MN%BP:(00155) Type=1.1 Nibs=3 Dist=001F7
                                    + 0EBEC MN%BP:(001FB) Type=1.1 Nibs=3 Dist=00151
                                    + 105B6 MN%CNF(00424) Type=1.1 Nibs=4 Dist=01879
                                    + 1060B MN%CNF(00479) Type=1.1 Nibs=4 Dist=018CE
                                    + 1C393 MB%USG(008EF) Type=0.1 Nibs=5
                                    + 1CA1A MN%CD:(001A1) Type=0.1 Nibs=5
                                    + 1CF8D MN%CD:(00714) Type=0.1 Nibs=5
CSLW9j =  1BA06 MB%IMG              - 17E9A SB%IO:(000DE) Type=1.0 Nibs=4 Dist=03B6C
                                    + 1BAC7 MB%USG(00023) Type=1.1 Nibs=3 Dist=000C1
CSLWP  =  09B63 TI%ERD              - 17E56 SB%IO:(0009A) Type=0.1 Nibs=5
CSLWP9 =  09B61 TI%ERD              - 1BA08 MB%IMG(005C2) Type=0.1 Nibs=5
CSPEED =  2F977 SB%RAM              - 0EA5A MN%BP:(00069) Type=0.0 Nibs=5
                                    + 0EB5F MN%BP:(0016E) Type=0.0 Nibs=5
CSRC1  =  1B427 AB%UTL              -
CSRC10 =  1B432 AB%UTL              - 0A791 SG%FXQ(00C1B) Type=0.1 Nibs=5
CSRC11 =  1B435 AB%UTL              -
CSRC12 =  1B438 AB%UTL              -
CSRC13 =  1B43B AB%UTL              -
CSRC14 =  1B43E AB%UTL              -
CSRC15 =  1B441 AB%UTL              -
CSRC2  =  1B424 AB%UTL              -
CSRC3  =  1B421 AB%UTL              - 105F5 MN%CNF(00463) Type=0.1 Nibs=5
CSRC4  =  1B41E AB%UTL              - 0AAA1 TI%FX2(0001C) Type=0.1 Nibs=5
CSRC5  =  1B41B AB%UTL              - 0F4CA AB%EXP(00352) Type=0.1 Nibs=5
CSRC6  =  1B418 AB%UTL              -
CSRC7  =  1B415 AB%UTL              -
CSRC8  =  1B42C AB%UTL              -
CSRC9  =  1B42F AB%UTL              -
CSRW3  =  0ED32 MN%UTL              - 04130 SB%EXP(0015B) Type=0.1 Nibs=5
                                    + 105D1 MN%CNF(0043F) Type=1.1 Nibs=4 Dist=0189F
                                    + 105E9 MN%CNF(00457) Type=1.1 Nibs=4 Dist=018B7
CSRW4  =  0ED2F MN%UTL              -
CSRW5  =  0ED2C MN%UTL              - 011B7 TI%UTL(00086) Type=0.1 Nibs=5
                                    + 0EBA8 MN%BP:(001B7) Type=1.1 Nibs=3 Dist=00184
                                    + 0EC02 MN%BP:(00211) Type=1.1 Nibs=3 Dist=0012A
                                    + 1049A MN%CNF(00308) Type=1.0 Nibs=4 Dist=0176E
                                    + 151BE MN%ED:(0053E) Type=1.1 Nibs=4 Dist=06492
                                    + 1D206 MN%CD:(0098D) Type=0.1 Nibs=5
CSRW9j =  1C07F MB%USG              - 17E73 SB%IO:(000B7) Type=1.1 Nibs=4 Dist=0420C
```

```
                                + 17EEF SB%IO:(00133) Type=1.1 Nibs=4 Dist=04190
                                + 1B74D MB%IMG(00307) Type=1.1 Nibs=4 Dist=00932
                                + 1B766 MB%IMG(00320) Type=1.1 Nibs=4 Dist=00919
CSRWP  =  09B4E TI%ERD          - 17E69 SB%IO:(000AD) Type=0.1 Nibs=5
CSRWP9 =  09B4C TI%ERD          - 1C081 MB%USG(005DD) Type=0.1 Nibs=5
CSTRST =  0EDCC MN%UTL          - 00250 SB%DVR(00250) Type=0.1 Nibs=5
CT$CRD =  1D7B6 MN%CD:          - 06738 SG%SYS(00398) Type=0.1 Nibs=5
CTRL   =  15285 MN%ED:          -
CUR020 =  10080 JP%MEM          - 07432 JP%SYS(00503) Type=0.1 Nibs=5
CURBOT =  10059 JP%MEM          - 01745 SB%CMD(0011E) Type=0.1 Nibs=5
CURDVC =  0A60B SG%FXQ          - 01133 TI%UTL(00002) Type=0.1 Nibs=5
                                + 06D17 SG%SYS(00977) Type=1.1 Nibs=4 Dist=038F4
                                + 07010 JP%SYS(000E1) Type=1.1 Nibs=4 Dist=035FB
CURREN =  2F56C SB%RAM          - 08524 JP%EXC(00748) Type=0.0 Nibs=5
                                + 0A51B SG%FXQ(009A5) Type=0.0 Nibs=5
                                + 1A680 SC%REN(00104) Type=0.0 Nibs=5
                                + 1A757 SC%REN(001DB) Type=0.0 Nibs=5
                                + 1A91D SC%REN(003A1) Type=0.0 Nibs=5
                                + 1A98F SC%REN(00413) Type=0.0 Nibs=5
CURRL  =  2F7E8 SB%RAM          - 069CF SG%SYS(0062F) Type=0.0 Nibs=5
                                + 0957D TI%ERD(001EA) Type=0.0 Nibs=5
                                + 10163 JP%MEM(00204) Type=0.0 Nibs=5
                                + 10180 JP%MEM(00221) Type=0.0 Nibs=5
CURRLO =  10161 JP%MEM          - 07421 JP%SYS(004F2) Type=0.1 Nibs=5
CURRST =  2F55D SB%RAM          - 06885 SG%SYS(004E5) Type=0.0 Nibs=5
                                + 06BF0 SG%SYS(00850) Type=0.0 Nibs=5
                                + 076F2 JP%SYS(007C3) Type=0.0 Nibs=5
                                + 07C14 JP%SYS(00CE5) Type=0.0 Nibs=5
                                + 0A4F5 SG%FXQ(0097F) Type=0.0 Nibs=5
                                + 19727 SC%SUB(009B5) Type=0.0 Nibs=5
                                + 1A2E7 SC%SUB(01575) Type=0.0 Nibs=5
                                + 1A38C SC%SUB(0161A) Type=0.0 Nibs=5
                                + 1A72E SC%REN(001B2) Type=0.0 Nibs=5
                                + 1DBAC TI%FX4(00002) Type=0.0 Nibs=5
CURSBj =  0173C SB%CMD          - 0049B SB%DVR(0049B) Type=1.0 Nibs=4 Dist=012A1
CURSDj =  0170F SB%CMD          - 0048F SB%DVR(0048F) Type=1.0 Nibs=4 Dist=01280
CURSFL =  151DF MN%ED:          - 0168E SB%CMD(00067) Type=0.1 Nibs=5
                                + 096C6 TI%ERD(00333) Type=0.1 Nibs=5
                                + 18AEE SB%IO:(00D32) Type=1.1 Nibs=4 Dist=0390F
CURSFR =  151D7 MN%ED:          - 003ED SB%DVR(003ED) Type=0.1 Nibs=5
                                + 18A50 SB%IO:(00C94) Type=1.1 Nibs=4 Dist=03879
CURSNP =  07412 JP%SYS          - 100DE JP%MEM(0017F) Type=0.1 Nibs=5
CURSOR =  2F47E SB%RAM          - 01BC7 SB%DSP(00337) Type=0.0 Nibs=5
                                + 020AC SB%DSP(0081C) Type=0.0 Nibs=5
                                + 02343 SB%DSP(00AB3) Type=0.0 Nibs=5
                                + 0234F SB%DSP(00ABF) Type=0.0 Nibs=4
                                + 151E9 MN%ED:(00569) Type=0.0 Nibs=5
                                + 15201 MN%ED:(00581) Type=0.0 Nibs=5
CURSRD =  100A4 JP%MEM          - 01718 SB%CMD(000F1) Type=0.1 Nibs=5
CURSRR =  151C7 MN%ED:          - 096D3 TI%ERD(00340) Type=0.1 Nibs=5
CURSRT =  096C1 TI%ERD          - 10144 JP%MEM(001E5) Type=1.1 Nibs=4 Dist=06A83
CURSRU =  1009A JP%MEM          - 016F1 SB%CMD(000CA) Type=0.1 Nibs=5
CURSTj =  01726 SB%CMD          - 00495 SB%DVR(00495) Type=1.0 Nibs=4 Dist=01291
CURSUj =  016E8 SB%CMD          - 00489 SB%DVR(00489) Type=1.0 Nibs=4 Dist=0125F
CURTOP =  10063 JP%MEM          - 0172F SB%CMD(00108) Type=0.1 Nibs=5
CVUCW  =  03FBC JP%PR3          - 01267 TI%UTL(00136) Type=1.0 Nibs=4 Dist=02D55
                                + 16D1E FH%TFM(00270) Type=0.1 Nibs=5
CkLoop =  1B669 MB%IMG          -
CkLpNC =  1B66D MB%IMG          -
Clear  =  00005 SB%DSP          - 004F1 SB%DVR(004F1) Type=0.0 Nibs=1
```

```
                                + 00591 SB%DVR(00591) Type=0.0 Nibs=1
Cslc5  =  0A924 SG%FXQ          - 0B089 AB%REG(00491) Type=1.1 Nibs=3 Dist=00765
                                + 0E22D PM%STA(003FE) Type=1.0 Nibs=4 Dist=03909
                                + 0F4B9 AB%EXP(00341) Type=1.1 Nibs=4 Dist=04B95
                                + 11777 SC%FIL(00724) Type=1.1 Nibs=4 Dist=06E53
                                + 11F93 SC%FIL(00F40) Type=1.1 Nibs=4 Dist=0766F
CurOff =  00006 SB%DSP          -

D'LTE  =  0695E SG%SYS          - 1EC11 SB%TAB(006DA) Type=1.2 Nibs=5 Dist=182B3
D'LTP  =  03AC8 JP%PR3          - 06959 SG%SYS(005B9) Type=1.2 Nibs=5 Dist=02E91
D0+2RD =  13A32 SC%DAT          -
D0=AVS =  09B2C TI%ERD          - 0FAD1 SC%TRC(0008A) Type=1.1 Nibs=4 Dist=05FA5
                                + 0FB44 SC%TRC(000FD) Type=1.1 Nibs=4 Dist=06018
                                + 17C83 SB%RD:(000BC) Type=0.1 Nibs=5
D0=CUR =  02093 SB%DSP          - 1521F MN%ED:(0059F) Type=0.1 Nibs=5
D0=FIB =  13AC5 SC%DAT          - 113AC SC%FIL(00359) Type=1.1 Nibs=4 Dist=02719
                                + 11407 SC%FIL(003B4) Type=1.1 Nibs=4 Dist=026BE
D0=OBS =  05067 SG%LDC          - 17261 FH%TFM(007B3) Type=0.1 Nibs=5
D0=PCA =  09B37 TI%ERD          - 07125 JP%SYS(001F6) Type=1.1 Nibs=4 Dist=02A12
                                + 07216 JP%SYS(002E7) Type=1.1 Nibs=4 Dist=02921
                                + 08774 SG%EXC(0012C) Type=1.1 Nibs=4 Dist=013C3
                                + 08A01 SG%EXC(003B9) Type=1.1 Nibs=4 Dist=01136
                                + 0FB07 SC%TRC(000C0) Type=1.1 Nibs=4 Dist=05FD0
                                + 0FE9B SC%TRC(00454) Type=1.1 Nibs=4 Dist=06364
                                + 0FEF7 SC%TRC(004B0) Type=1.1 Nibs=4 Dist=063C0
                                + 1161E SC%FIL(005CB) Type=1.1 Nibs=4 Dist=07AE7
D0ASC+ =  0982C TI%ERD          -
D0ASCI =  09833 TI%ERD          - 0512F SG%LDC(001D1) Type=1.0 Nibs=4 Dist=04704
                                + 068E8 SG%SYS(00548) Type=1.1 Nibs=4 Dist=02F4B
D0OUTB =  05067 SG%LDC          - 06813 SG%SYS(00473) Type=1.1 Nibs=4 Dist=017AC
                                + 09718 TI%ERD(00385) Type=1.1 Nibs=4 Dist=046B1
                                + 0FF79 JP%MEM(0001A) Type=0.1 Nibs=5
D12ROA =  1BA3C MB%IMG          - 1C1B6 MB%USG(00712) Type=1.1 Nibs=3 Dist=0077A
D1=AVE =  18651 SB%IO:          - 066E5 SG%SYS(00345) Type=0.1 Nibs=5
                                + 0680C SG%SYS(0046C) Type=0.1 Nibs=5
                                + 0E49B PM%STA(0066C) Type=0.1 Nibs=5
                                + 11F6A SC%FIL(00F17) Type=1.1 Nibs=4 Dist=066E7
                                + 17C99 SB%RD:(000D2) Type=1.1 Nibs=4 Dist=009B8
                                + 17D6B SB%RD:(001A4) Type=1.1 Nibs=4 Dist=008E6
                                + 1BA02 MB%IMG(005BC) Type=1.0 Nibs=4 Dist=033B1
D1=CRS =  06883 SG%SYS          - 0113E TI%UTL(0000D) Type=1.1 Nibs=4 Dist=05745
                                + 0A1AB SG%FXQ(00635) Type=1.1 Nibs=4 Dist=03928
                                + 0A60D SG%FXQ(00A97) Type=1.1 Nibs=4 Dist=03D8A
                                + 0FE45 SC%TRC(003FE) Type=0.1 Nibs=5
                                + 0FFB1 JP%MEM(00052) Type=0.1 Nibs=5
                                + 16D3F FH%TFM(00291) Type=0.1 Nibs=5
D1=IBS =  01472 TI%UTL          - 0264D JP%PR1(000AD) Type=1.1 Nibs=4 Dist=011DB
D1@AVS =  01299 TI%UTL          - 176C3 FH%TFM(00C15) Type=0.1 Nibs=5
D1C=R3 =  03047 JP%PR2          - 02A5A JP%PR1(004BA) Type=1.1 Nibs=3 Dist=005ED
                                + 03646 JP%PR3(00270) Type=1.1 Nibs=3 Dist=005FF
                                + 036AE JP%PR3(002D8) Type=1.1 Nibs=3 Dist=00667
                                + 03D6E JP%PR3(00998) Type=1.1 Nibs=4 Dist=00D27
                                + 051FD SG%LDC(0029F) Type=1.1 Nibs=4 Dist=021B6
                                + 09808 TI%ERD(00475) Type=1.1 Nibs=4 Dist=067C1
D1FSTK =  1955D SC%SUB          - 088AF SG%EXC(00267) Type=0.1 Nibs=5
                                + 089F2 SG%EXC(003AA) Type=0.1 Nibs=5
                                + 11429 SC%FIL(003D6) Type=0.1 Nibs=5
D1MST+ =  13E21 SC%DAT          - 17BFE SB%RD:(00037) Type=1.1 Nibs=4 Dist=03DDD
D1MSTK =  1954E SC%SUB          - 145FE SC%DAT(00D49) Type=1.0 Nibs=4 Dist=04F50
                                + 15DCE AB%CLC(008AF) Type=1.1 Nibs=4 Dist=03780
```

```
                                    + 16339 AB%BLD(0000D) Type=1.1 Nibs=4 Dist=03215
D1STOR =  18BA9 SB%IO:              - 0E4A9 PM%STA(0067A) Type=0.1 Nibs=5
                                    + 17D65 SB%RD:(0019E) Type=1.1 Nibs=4 Dist=00E44
D1mstk =  145FC SC%DAT              - 0FD8D SC%TRC(00346) Type=1.1 Nibs=4 Dist=0486F
                                    + 0FDE9 SC%TRC(003A2) Type=1.1 Nibs=4 Dist=04813
D=AVME =  1A476 SB%EXC              - 01760 SB%CMD(00139) Type=0.1 Nibs=5
                                    + 03D5F JP%PR3(00989) Type=0.1 Nibs=5
                                    + 03FDE SB%EXP(00009) Type=0.1 Nibs=5
                                    + 04755 SB%EXP(00780) Type=0.1 Nibs=5
                                    + 05062 SG%LDC(00104) Type=0.1 Nibs=5
                                    + 057D0 SG%LDC(00872) Type=0.1 Nibs=5
                                    + 0FACA SC%TRC(00083) Type=0.1 Nibs=5
                                    + 10879 MN%CNF(006E7) Type=0.1 Nibs=5
                                    + 1AB07 SG%KEY(00066) Type=1.1 Nibs=3 Dist=00691
D=AVMS =  1A460 SB%EXC              - 0A99E SB%FCN(00073) Type=0.1 Nibs=5
                                    + 0A9FA SB%FCN(000CF) Type=0.1 Nibs=5
                                    + 18695 SB%IO:(008D9) Type=1.0 Nibs=4 Dist=01DCB
                                    + 1ACB0 SG%KEY(0020F) Type=1.1 Nibs=4 Dist=00850
                                    + 1B50A MB%IMG(000C4) Type=1.1 Nibs=4 Dist=010AA
                                    + 1B9FC MB%IMG(005B6) Type=1.1 Nibs=4 Dist=0159C
D=WORD =  04C0E AB%LEX              -
DATA   =  08A20 SG%EXC              - 1EC98 SB%TAB(00761) Type=1.2 Nibs=5 Dist=16278
DATACK =  036DE JP%PR3              - 03201 JP%PR2(005DB) Type=1.1 Nibs=3 Dist=004DD
DATADC =  05352 SG%LDC              - 08A16 SG%EXC(003CE) Type=1.2 Nibs=5 Dist=036C4
DATCOD =  1483A SC%DAT              - 1484A SC%DAT(00F95) Type=0.0 Nibs=5
DATE   =  12BD4 MN%TM:              - 1E9D1 SB%TAB(0049A) Type=1.2 Nibs=5 Dist=0BDFD
DATE$  =  12C3A MN%TM:              - 1E9DA SB%TAB(004A3) Type=1.2 Nibs=5 Dist=0BDA0
DATLEN =  0B584 AB%REG              - 0FDA4 SC%TRC(0035D) Type=1.1 Nibs=4 Dist=04820
                                    + 1485C SC%DAT(00FA7) Type=0.1 Nibs=5
                                    + 14922 SC%DAT(0106D) Type=0.1 Nibs=5
DATP   =  03568 JP%PR3              - 08A1B SG%EXC(003D3) Type=1.2 Nibs=5 Dist=054B3
DATPTR =  2F692 SB%RAM              - 07AD0 JP%SYS(00BA1) Type=0.0 Nibs=5
                                    + 17C34 SB%RD:(0006D) Type=0.0 Nibs=4
                                    + 17CE3 SB%RD:(0011C) Type=0.0 Nibs=5
                                    + 17D54 SB%RD:(0018D) Type=0.0 Nibs=5
DAY2JD =  13407 MN%TM:              -
DAYYMD =  13335 MN%TM:              - 1C911 MN%CD:(00098) Type=0.1 Nibs=5
DBLPI4 =  0DAFC SM%MTH              -
DBLSUB =  0DADD SM%MTH              - 0CFE7 JT%MTH(00CC0) Type=1.1 Nibs=4 Dist=00AF6
DCHX=C =  1B2D0 AB%UTL              - 1AD36 SG%KEY(00295) Type=1.1 Nibs=3 Dist=0059A
                                    + 1AD41 SG%KEY(002A0) Type=1.1 Nibs=3 Dist=0058F
DCHXF  =  1B223 AB%UTL              - 0E296 PM%STA(00467) Type=0.1 Nibs=5
                                    + 136D3 PM%FLG(0028A) Type=1.1 Nibs=4 Dist=07B50
DCHXW  =  0ECDC MN%UTL              - 0EB11 MN%BP:(00120) Type=1.1 Nibs=3 Dist=001CB
                                    + 0EB31 MN%BP:(00140) Type=1.1 Nibs=3 Dist=001AB
                                    + 12CE8 MN%TM:(0079E) Type=1.0 Nibs=4 Dist=0400C
DCONTR =  2E3FE SB%RAM              - 0F08D MN%UTL(0041F) Type=0.0 Nibs=5
DCPLIN =  10108 JP%MEM              - 073E6 JP%SYS(004B7) Type=0.1 Nibs=5
DCRMNT =  1C177 MB%USG              -
DD1CTL =  2E3FF SB%RAM              - 005BF SB%DVR(005BF) Type=0.0 Nibs=4
                                    + 00684 SB%DVR(00684) Type=0.0 Nibs=2 Offset=     -1
                                    + 0198C SB%DSP(000FC) Type=0.0 Nibs=2
                                    + 01AB0 SB%DSP(00220) Type=0.0 Nibs=5
                                    + 0EF2D MN%UTL(002BF) Type=0.0 Nibs=5
DD1END =  2E34C SB%RAM              - 1C401 SB%GPH(0003C) Type=0.0 Nibs=5
                                    + 1C498 SB%GPH(000D3) Type=0.0 Nibs=2
DD1ST  =  2E300 SB%RAM              - 1C492 SB%GPH(000CD) Type=0.0 Nibs=4
DD2CTL =  2E2FF SB%RAM              -
DD2END =  2E260 SB%RAM              - 1C40E SB%GPH(00049) Type=0.0 Nibs=5
DD2ST  =  2E200 SB%RAM              - 1C488 SB%GPH(000C3) Type=0.0 Nibs=4
```

```
DD3CTL =  2E1FF SB%RAM       - 005D3 SB%DVR(005D3) Type=0.0 Nibs=2
DD3END =  2E160 SB%RAM       - 1C41F SB%GPH(0005A) Type=0.0 Nibs=5
                             + 1C480 SB%GPH(000BB) Type=0.0 Nibs=2
DD3ST  =  2E104 SB%RAM       - 01A26 SB%DSP(00196) Type=0.0 Nibs=5
                             + 1C479 SB%GPH(000B4) Type=0.0 Nibs=5
DE-REF =  170D8 FH%TFM       -
DEBNCE =  00CF7 SB%KEY       - 0013B SB%DVR(0013B) Type=0.1 Nibs=5
                             + 006C3 SB%DVR(006C3) Type=1.1 Nibs=3 Dist=00634
                             + 072F0 JP%SYS(003C1) Type=1.1 Nibs=4 Dist=065F9
                             + 152E1 MN%ED:(00661) Type=0.1 Nibs=5
DECDC  =  05287 SG%LDC       - 0ABF8 AB%REG(00000) Type=1.2 Nibs=5 Dist=05971
                             + 0AC06 AB%REG(0000E) Type=1.2 Nibs=5 Dist=0597F
DECHEX =  1B2D2 AB%UTL       - 08EBE SG%EXC(00876) Type=0.1 Nibs=5
                             + 09E3C SG%FXQ(002C6) Type=0.1 Nibs=5
DECP   =  0328F JP%PR2       - 0ABFD AB%REG(00005) Type=1.2 Nibs=5 Dist=0796E
DECP=C =  0C751 JT%MTH       - 1BDA4 MB%USG(00300) Type=0.1 Nibs=5
DEF    =  198F9 SC%SUB       - 1EC23 SB%TAB(006EC) Type=1.2 Nibs=5 Dist=0532A
DEFADC =  052FC SG%LDC       - 13770 PM%FLG(00327) Type=1.2 Nibs=5 Dist=0E474
DEFADR =  2F967 SB%RAM       - 14CB1 MN%ED:(00031) Type=0.0 Nibs=5
                             + 14DAC MN%ED:(0012C) Type=0.0 Nibs=5 Offset=      3
                             + 14FD0 MN%ED:(00350) Type=0.0 Nibs=5
                             + 156AD AB%CLC(0018E) Type=0.0 Nibs=5
                             + 15DBF AB%CLC(008A0) Type=0.0 Nibs=5
                             + 169EE AB%ED:(0037C) Type=0.0 Nibs=5
DEFALT =  1377A PM%FLG       - 1DE6B JP%TAB(00167) Type=1.2 Nibs=5 Dist=0A6F1
DEFAP  =  03CA4 JP%PR3       - 13775 PM%FLG(0032C) Type=1.2 Nibs=5 Dist=0FAD1
DEFDC  =  05216 SG%LDC       - 198EF SC%SUB(00B7D) Type=1.2 Nibs=5 Dist=146D9
DEFEND =  07627 JP%SYS       - 0B24A AB%REG(00652) Type=1.0 Nibs=4 Dist=03C23
DEFFIL =  01131 TI%UTL       - 081F8 JP%EXC(0041C) Type=1.1 Nibs=4 Dist=070C7
                             + 16B1C FH%TFM(0006E) Type=0.1 Nibs=5
DEFP   =  03093 JP%PR2       - 198F4 SC%SUB(00B82) Type=1.2 Nibs=5 Dist=16861
DEFPRM =  18B15 SB%IO:       - 1894D SB%IO:(00B91) Type=0.0 Nibs=5
                             + 18A14 SB%IO:(00C58) Type=0.0 Nibs=5
DEG    =  0C155 AB%FCN       - 1E989 SB%TAB(00452) Type=1.2 Nibs=5 Dist=12834
DEGREE =  13863 PM%FLG       - 1ED0D SB%TAB(007D6) Type=1.2 Nibs=5 Dist=0B4AA
DELAY  =  0EEA6 MN%UTL       - 1ED28 SB%TAB(007F1) Type=1.2 Nibs=5 Dist=0FE82
DELAYT =  2F948 SB%RAM       - 01DDB SB%DSP(0054B) Type=0.0 Nibs=5
                             + 0EECF MN%UTL(00261) Type=0.0 Nibs=5
DELAYd =  05493 SG%LDC       - 0EE9C MN%UTL(0022E) Type=1.2 Nibs=5 Dist=09A09
                             + 18C58 MN%LCK(00000) Type=1.2 Nibs=5 Dist=137C5
                             + 1A484 SB%EXC(000B0) Type=1.2 Nibs=5 Dist=14FF1
                             + 1A4C4 SB%EXC(000F0) Type=1.2 Nibs=5 Dist=15031
                             + 1C458 SB%GPH(00093) Type=1.2 Nibs=5 Dist=16FC5
DELAYp =  02AC6 JP%PR1       - 0EEA1 MN%UTL(00233) Type=1.2 Nibs=5 Dist=0C3DB
DELFIB =  12095 SC%FIL       -
DEST   =  0F7B0 AB%ASN       - 0E493 PM%STA(00664) Type=1.1 Nibs=4 Dist=0131D
                             + 13E5E SC%DAT(005A9) Type=1.1 Nibs=4 Dist=046AE
                             + 15E2A AB%CLC(0090B) Type=1.1 Nibs=4 Dist=0667A
                             + 18F3F SC%SUB(001CD) Type=0.1 Nibs=5
DEST+  =  18C42 SB%IO:       -
DFKEYP =  03754 JP%PR3       - 03099 JP%PR2(00473) Type=1.2 Nibs=3 Dist=006BB
DFLTCR =  0F857 AB%ASN       - 193A6 SC%SUB(00634) Type=0.1 Nibs=5
DIM    =  0AC10 AB%REG       - 1ECCE SB%TAB(00797) Type=1.2 Nibs=5 Dist=140BE
DIM1O  =  0F85F AB%ASN       - 0BA83 AB%FCN(0044A) Type=0.0 Nibs=5 Offset=     -3
DIMP   =  032BB JP%PR2       - 0AC0B AB%REG(00013) Type=1.2 Nibs=5 Dist=07950
DIMSTM =  0AC10 AB%REG       -
DISINT =  2F470 SB%RAM       - 0007A SB%DVR(0007A) Type=0.0 Nibs=5
                             + 00DE8 SB%KEY(000F1) Type=0.0 Nibs=2
                             + 00F78 SB%KEY(00281) Type=0.0 Nibs=5
DISP   =  17F33 SB%IO:       - 1EC8F SB%TAB(00758) Type=1.2 Nibs=5 Dist=06D5C
```

```
DISPDC =  05450 SG%LDC        - 17F29 SB%IO:(0016D) Type=1.2 Nibs=5 Dist=12AD9
DISPP  =  035A4 JP%PR3        - 02F49 JP%PR2(00323) Type=1.1 Nibs=3 Dist=0065B
                              + 17F2E SB%IO:(00172) Type=1.2 Nibs=5 Dist=1498A
DISPt  =  00000 TI%EQU        -
DIV    =  0C143 AB%FCN        - 1EA58 SB%TAB(00521) Type=1.2 Nibs=5 Dist=12915
DIVO   =  0C4C6 JT%MTH        -
DIV100 =  0C4FA JT%MTH        - 0DDF4 SM%MTH(0098F) Type=1.1 Nibs=4 Dist=018FA
DIV120 =  0C50F JT%MTH        - 0DD80 SM%MTH(0094B) Type=1.1 Nibs=4 Dist=01871
DIV15  =  0C505 JT%MTH        -
DIVF   =  0C4B8 JT%MTH        -
DIVFCD =  0C4BB JT%MTH        - 0D9A3 SM%MTH(0056E) Type=1.1 Nibs=4 Dist=014E8
DIVIDE =  0B685 AB%FCN        - 1EA46 SB%TAB(0050F) Type=1.2 Nibs=5 Dist=133C1
DLFIBA =  12110 SC%FIL        - 070ED JP%SYS(001BE) Type=0.1 Nibs=5
                              + 07248 JP%SYS(00319) Type=0.1 Nibs=5
DMARRY =  0BBB2 AB%FCN        - 1EA10 SB%TAB(004D9) Type=1.2 Nibs=5 Dist=12E5E
DMNSN  =  0AE39 AB%REG        -
DNAT12 =  0628B TI%FX1        - 06136 SB%EXD(00814) Type=1.0 Nibs=3 Dist=00155
DNAT18 =  06139 SB%EXD        - 062A1 TI%FX1(00051) Type=1.0 Nibs=3 Dist=00168
DONNA  =  09656 TI%ERD        - 18A35 SB%IO:(00C79) Type=0.1 Nibs=5
DOSCRL =  01CDF SB%DSP        - 17EFF SB%IO:(00143) Type=0.1 Nibs=5
DPART2 =  17EA3 SB%IO:        - 18574 SB%IO:(007B8) Type=0.0 Nibs=5
                              + 185A6 SB%IO:(007EA) Type=0.0 Nibs=5
DPART3 =  17EF8 SB%IO:        - 17E9E SB%IO:(000E2) Type=1.2 Nibs=5 Dist=0005A
DPOS   =  2F94D SB%RAM        - 022A4 SB%DSP(00A14) Type=0.0 Nibs=5
                              + 18565 SB%IO:(007A9) Type=0.0 Nibs=5
DPVCTR =  0AC50 AB%REG        -
DRANGE =  1B076 AB%UTL        - 0384A JP%PR3(00474) Type=0.1 Nibs=5
                              + 03F7A JP%PR3(00BA4) Type=0.1 Nibs=5
                              + 04614 SB%EXP(0063F) Type=0.1 Nibs=5
                              + 04CB0 AB%LEX(003CA) Type=0.1 Nibs=5
                              + 05A4B SB%EXD(00129) Type=0.1 Nibs=5
                              + 08EA9 SG%EXC(00861) Type=0.1 Nibs=5
                              + 09E52 SG%FXQ(002DC) Type=0.1 Nibs=5
                              + 0FB68 SC%TRC(00121) Type=0.1 Nibs=5
                              + 12F71 MN%TM:(00A27) Type=0.1 Nibs=5
                              + 1627E AB%CLC(00D5F) Type=1.1 Nibs=4 Dist=04DF8
                              + 1B674 MB%IMG(0022E) Type=1.1 Nibs=3 Dist=005FE
                              + 1C5E5 SG%POK(000F7) Type=1.1 Nibs=4 Dist=0156F
DROP   =  0DE39 PM%STA        - 1DE74 JP%TAB(00170) Type=1.2 Nibs=5 Dist=1003B
DROPDC =  05470 SG%LDC        - 0DE2F PM%STA(00000) Type=1.2 Nibs=5 Dist=089BF
                              + 1A3DA SB%EXC(00006) Type=1.2 Nibs=5 Dist=14F6A
DROPP  =  03A03 JP%PR3        - 0DE34 PM%STA(00005) Type=1.2 Nibs=5 Dist=0A431
DSLEEP =  0056D SB%DVR        - 07FEF JP%EXC(00213) Type=1.1 Nibs=4 Dist=07A82
                              + 166F1 AB%ED:(0007F) Type=0.1 Nibs=5
                              + 18B4F SB%IO:(00D93) Type=0.1 Nibs=5
DSP$   =  185D8 SB%IO:        - 1E036 JP%TAB(00332) Type=1.2 Nibs=5 Dist=05A5E
DSP$00 =  185DB SB%IO:        - 18D11 MN%LCK(000B9) Type=1.1 Nibs=3 Dist=00736
DSPBFE =  2F540 SB%RAM        - 163AA AB%BLD(0007E) Type=0.0 Nibs=5
DSPBFS =  2F480 SB%RAM        - 0177C SB%CMD(00155) Type=0.0 Nibs=4
                              + 01793 SB%CMD(0016C) Type=0.0 Nibs=4 Offset=     96
                              + 018E6 SB%DSP(00056) Type=0.0 Nibs=5 Offset=     -5
                              + 01956 SB%DSP(000C6) Type=0.0 Nibs=5
                              + 01B2F SB%DSP(0029F) Type=0.0 Nibs=5
                              + 01D5E SB%DSP(004CE) Type=0.0 Nibs=5
                              + 020A5 SB%DSP(00815) Type=0.0 Nibs=5
                              + 020D0 SB%DSP(00840) Type=0.0 Nibs=5
                              + 02255 SB%DSP(009C5) Type=0.0 Nibs=5
                              + 16420 AB%BLD(000F4) Type=0.0 Nibs=5
                              + 16622 AB%BLD(002F6) Type=0.0 Nibs=4
                              + 1688A AB%ED:(00218) Type=0.0 Nibs=5 Offset=     -2
```

```
                                + 1694C ABXED:(002DA) Type=0.0 Nibs=5
                                + 169C2 ABXED:(00350) Type=0.0 Nibs=5
                                + 185F7 SBXIO:(0083B) Type=0.0 Nibs=4
                                + 1860E SBXIO:(00852) Type=0.0 Nibs=4 Offset=     96
DSPBUF =  09723 TIXERD          -
DSPCHA =  01C3E SBXDSP          - 00142 SBXDVR(00142) Type=0.1 Nibs=5
                                + 0257A MBXROM(000F6) Type=1.1 Nibs=4 Dist=0093C
                                + 096B6 TIXERD(00323) Type=1.1 Nibs=4 Dist=07A78
                                + 09756 TIXERD(003C3) Type=1.1 Nibs=4 Dist=07B18
                                + 14DF4 MNXED:(00174) Type=0.1 Nibs=5
                                + 17EDB SBXIO:(0011F) Type=0.1 Nibs=5
                                + 1D84D MNXCD:(00FD4) Type=0.1 Nibs=5
DSPCHC =  01C3C SBXDSP          - 0105B SBXKEY(00364) Type=1.1 Nibs=4 Dist=00BE1
                                + 151FA MNXED:(0057A) Type=0.1 Nibs=5
DSPCHX =  2F674 SBXRAM          - 01028 SBXKEY(00331) Type=0.0 Nibs=5
                                + 01C5F SBXDSP(003CF) Type=0.0 Nibs=5
                                + 10EAE MNXCNF(00D1C) Type=0.0 Nibs=5
DSPCL? =  020B6 SBXDSP          - 14CA6 MNXED:(00026) Type=0.1 Nibs=5
DSPCNA =  09721 TIXERD          - 01687 SBXCMD(00060) Type=0.1 Nibs=5
                                + 0FDE3 SCXTRC(0039C) Type=1.1 Nibs=4 Dist=066C2
                                + 16352 ABXBLD(00026) Type=0.1 Nibs=5
DSPCNB =  0971F TIXERD          -
DSPCNO =  09716 TIXERD          - 072C9 JPXSYS(0039A) Type=1.1 Nibs=4 Dist=0244D
                                + 1013B JPXMEM(001DC) Type=1.1 Nibs=4 Dist=06A25
DSPDGT =  2F6DD SBXRAM          - 0F04F MNXUTL(003E1) Type=0.0 Nibs=5
DSPDLY =  01DAF SBXDSP          - 06443 SGXSYS(000A3) Type=1.1 Nibs=4 Dist=04694
                                + 1D8F8 MNXCD:(0107F) Type=0.1 Nibs=5
DSPF   =  0F00C MNXUTL          - 1DDDB JPXTAB(000D7) Type=1.2 Nibs=5 Dist=0EDCF
                                + 1DDE4 JPXTAB(000E0) Type=1.2 Nibs=5 Dist=0EDD8
                                + 1DDED JPXTAB(000E9) Type=1.2 Nibs=5 Dist=0EDE1
DSPFMT =  2F6DC SBXRAM          - 0F026 MNXUTL(003B8) Type=0.0 Nibs=5
                                + 18259 SBXIO:(0049D) Type=0.0 Nibs=5
DSPLI+ =  1010F JPXMEM          - 1ABC9 SGXKEY(00128) Type=0.1 Nibs=5
DSPLIN =  10127 JPXMEM          - 06A1A SGXSYS(0067A) Type=0.1 Nibs=5
DSPMSK =  2F540 SBXRAM          - 01771 SBXCMD(0014A) Type=0.0 Nibs=5 Offset=     12
                                + 01789 SBXCMD(00162) Type=0.0 Nibs=4
                                + 01BBC SBXDSP(0032C) Type=0.0 Nibs=5 Offset=     23
                                + 0211C SBXDSP(0088C) Type=0.0 Nibs=4
                                + 022DE SBXDSP(00A4E) Type=0.0 Nibs=5 Offset=     23
                                + 1023D MNXCNF(000AB) Type=0.0 Nibs=5 Offset=     14
                                + 185EC SBXIO:(00830) Type=0.0 Nibs=5 Offset=     12
                                + 18604 SBXIO:(00848) Type=0.0 Nibs=4
                                + 18A3C SBXIO:(00C80) Type=0.0 Nibs=5
                                + 18A56 SBXIO:(00C9A) Type=0.0 Nibs=4
DSPPO2 =  035A7 JPXPR3          - 02970 JPXPR1(003D0) Type=1.1 Nibs=4 Dist=00C37
DSPRST =  02443 SBXDSP          - 00149 SBXDVR(00149) Type=0.1 Nibs=5
                                + 01041 SBXKEY(0034A) Type=1.1 Nibs=4 Dist=01402
                                + 15564 ABXCLC(00045) Type=0.1 Nibs=5
                                + 1632E ABXBLD(00002) Type=0.1 Nibs=5
                                + 1687C ABXED:(0020A) Type=0.1 Nibs=5
                                + 168B1 ABXED:(0023F) Type=0.1 Nibs=5
DSPSET =  2F7B1 SBXRAM          -
DSPSTA =  2F475 SBXRAM          - 01B82 SBXDSP(002F2) Type=0.0 Nibs=5
                                + 01B95 SBXDSP(00305) Type=0.0 Nibs=5
                                + 01BA2 SBXDSP(00312) Type=0.0 Nibs=5 Offset=      3
                                + 01BAF SBXDSP(0031F) Type=0.0 Nibs=5 Offset=      3
                                + 02445 SBXDSP(00BB5) Type=0.0 Nibs=5 Offset=      3
                                + 16719 ABXED:(000A7) Type=0.0 Nibs=5 Offset=      3
                                + 168D7 ABXED:(00265) Type=0.0 Nibs=4 Offset=      3
DSPUP1 =  01ADE SBXDSP          - 00741 SBXDVR(00741) Type=1.1 Nibs=4 Dist=0139D
```

```
DSPUPD =  01ADA SB%DSP        - 00150 SB%DVR(00150) Type=0.1 Nibs=5
DSTA30 =  0B24E AB%REG        -
DSTRDC =  05280 SG%LDC        - 0B210 AB%REG(00618) Type=1.2 Nibs=5 Dist=05F90
DSTROY =  0B21A AB%REG        - 1EC50 SB%TAB(00719) Type=1.2 Nibs=5 Dist=13A36
DSTRY* =  0B0C9 AB%REG        -
DSTRYP =  03318 JP%PR2        - 0B215 AB%REG(0061D) Type=1.2 Nibs=5 Dist=07EFD
DSTp   =  0325A JP%PR2        - 039DD JP%PR3(00607) Type=1.1 Nibs=3 Dist=00783
                              + 039ED JP%PR3(00617) Type=1.1 Nibs=3 Dist=00793
DTR    =  0C181 AB%FCN        -
DUMA80 =  06250 TI%FX1        - 048D3 SB%EXP(008FE) Type=1.1 Nibs=4 Dist=0197D
DV15M  =  0C4AC JT%MTH        - 0C165 AB%FCN(00B2C) Type=1.1 Nibs=3 Dist=00347
DV15S  =  0C4B2 JT%MTH        - 0E504 PM%STA(006D5) Type=1.1 Nibs=4 Dist=02052
DV2-12 =  0C4A8 JT%MTH        - 0B68E AB%FCN(00055) Type=1.1 Nibs=4 Dist=00E1A
                              + 12BB4 MN%TM:(0066A) Type=1.1 Nibs=4 Dist=0670C
DV2-15 =  0C4AC JT%MTH        - 0D6B9 SM%MTH(00284) Type=1.1 Nibs=4 Dist=0120D
DVCNFE =  0A459 SG%FXQ        - 06728 SG%SYS(00388) Type=1.0 Nibs=4 Dist=03D31
                              + 17BC2 MN%FRP(0018F) Type=0.1 Nibs=5
DVZ    =  1389A PM%FLG        - 1EBDB SB%TAB(006A4) Type=1.2 Nibs=5 Dist=0B341
DVZNIB =  2F6FC SB%RAM        -
DWIDTH =  2F94F SB%RAM        - 00295 SB%DVR(00295) Type=0.0 Nibs=4
                              + 0EFCE MN%UTL(00360) Type=0.0 Nibs=5
DXP100 =  0CF7F JT%MTH        -
DYNAMC =  0F276 AB%EXP        - 1E902 SB%TAB(003CB) Type=1.2 Nibs=5 Dist=0F68C
                              + 1E90B SB%TAB(003D4) Type=1.2 Nibs=5 Dist=0F695
                              + 1E914 SB%TAB(003DD) Type=1.2 Nibs=5 Dist=0F69E
                              + 1E91D SB%TAB(003E6) Type=1.2 Nibs=5 Dist=0F6A7
                              + 1E926 SB%TAB(003EF) Type=1.2 Nibs=5 Dist=0F6B0
                              + 1E92F SB%TAB(003F8) Type=1.2 Nibs=5 Dist=0F6B9
                              + 1E938 SB%TAB(00401) Type=1.2 Nibs=5 Dist=0F6C2
                              + 1E941 SB%TAB(0040A) Type=1.2 Nibs=5 Dist=0F6CB
                              + 1E94A SB%TAB(00413) Type=1.2 Nibs=5 Dist=0F6D4
                              + 1E953 SB%TAB(0041C) Type=1.2 Nibs=5 Dist=0F6DD
DZ10   =  0C4DE JT%MTH        -
DZINF  =  0C4E4 JT%MTH        - 0D5F5 SM%MTH(001C0) Type=1.1 Nibs=4 Dist=01111
DZP    =  00003 JT%MTH        - 0D9BB SM%MTH(00586) Type=0.0 Nibs=1

EDIT   =  0A52C SG%FXQ        - 1EC1A SB%TAB(006E3) Type=1.2 Nibs=5 Dist=146EE
EDIT20 =  0A4E6 SG%FXQ        - 070BA JP%SYS(0018B) Type=1.1 Nibs=4 Dist=0342C
EDIT80 =  0A5A5 SG%FXQ        - 0657A SG%SYS(001DA) Type=1.1 Nibs=4 Dist=0402B
EDITP  =  03B8C JP%PR3        - 0A527 SG%FXQ(009B1) Type=1.2 Nibs=5 Dist=0699B
EDITWF =  0A533 SG%FXQ        - 002BE SB%DVR(002BE) Type=0.1 Nibs=5
                              + 10E6A MN%CNF(00CD8) Type=1.1 Nibs=4 Dist=06937
                              + 16D6B FH%TFM(002BD) Type=0.1 Nibs=5
EFIELD =  00000 JT%MTH        -
ELSE   =  091CA SG%EXC        - 1EE3F SB%TAB(00908) Type=1.2 Nibs=5 Dist=15C75
ELSEP  =  03491 JP%PR3        - 02819 JP%PR1(00279) Type=1.1 Nibs=4 Dist=00C78
END    =  0763F JP%SYS        - 18D7E SC%SUB(0000C) Type=0.1 Nibs=5
                              + 1ED4C SB%TAB(00815) Type=1.2 Nibs=5 Dist=1770D
END10  =  0764B JP%SYS        - 195AA SC%SUB(00838) Type=0.1 Nibs=5
END20  =  07664 JP%SYS        -
ENDALL =  0769A JP%SYS        - 06D7F SG%SYS(009DF) Type=1.0 Nibs=4 Dist=00918
                              + 0A068 SG%FXQ(004F2) Type=1.0 Nibs=4 Dist=029CE
                              + 16B6B FH%TFM(000BD) Type=0.1 Nibs=5
ENDBIN =  0764B JP%SYS        -
ENDDC  =  052F8 SG%LDC        - 07635 JP%SYS(00706) Type=1.2 Nibs=5 Dist=0233D
ENDDDC =  055C8 SG%LDC        - 19EA2 SC%SUB(01130) Type=1.2 Nibs=5 Dist=148DA
ENDDEF =  19EAC SC%SUB        - 07629 JP%SYS(006FA) Type=0.1 Nibs=5
                              + 1EC2C SB%TAB(006F5) Type=1.2 Nibs=5 Dist=04D80
ENDFIL =  11277 SC%FIL        -
ENDIMG =  1C040 MB%USG        -
```

```
ENDLIN =  1A3E4 SB%EXC        - 1DE86 JP%TAB(00182) Type=1.2 Nibs=5 Dist=03AA2
ENDP   =  03A24 JP%PR3        - 0763A JP%SYS(0070B) Type=1.2 Nibs=5 Dist=03C16
ENDSB- =  195AF SC%SUB        - 07630 JP%SYS(00701) Type=0.1 Nibs=5
ENDSDC =  055A8 SG%LDC        - 1959E SC%SUB(0082C) Type=1.2 Nibs=5 Dist=13FF6
ENDSUB =  195A8 SC%SUB        - 1EC74 SB%TAB(0073D) Type=1.2 Nibs=5 Dist=056CC
EOFCHK =  170FF FH%TFM        -
EOFIL  =  13FEA SC%DAT        -
EOFLC+ =  06EEE SG%SYS        - 0127C TI%UTL(0014B) Type=1.1 Nibs=4 Dist=05C72
                              + 08580 JP%EXC(007A4) Type=1.1 Nibs=4 Dist=01692
                              + 0BFB2 AB%FCN(00979) Type=1.1 Nibs=4 Dist=050C4
                              + 14AB6 SC%DAT(01201) Type=0.1 Nibs=5
EOFLC1 =  06EF1 SG%SYS        - 0A637 SG%FXQ(00AC1) Type=1.1 Nibs=4 Dist=03746
EOFLCH =  06EEA SG%SYS        - 0A12F SG%FXQ(005B9) Type=1.1 Nibs=4 Dist=03245
EOLCK  =  02A7E JP%PR1        - 02EDB JP%PR2(002B5) Type=1.1 Nibs=3 Dist=0045D
EOLCK+ =  02A7E JP%PR1        - 02978 JP%PR1(003D8) Type=1.1 Nibs=3 Dist=00106
                              + 02E49 JP%PR2(00223) Type=1.1 Nibs=3 Dist=003CB
                              + 03BF9 JP%PR3(00823) Type=1.1 Nibs=4 Dist=0117B
EOLCKR =  02A7A JP%PR1        - 035CC JP%PR3(001F6) Type=1.1 Nibs=4 Dist=00B52
EOLDC  =  05402 SG%LDC        -
EOLLEN =  2F95A SB%RAM        - 185AF SB%IO:(007F3) Type=0.0 Nibs=5 Offset=     -5
                              + 1A3F2 SB%EXC(0001E) Type=0.0 Nibs=5
                              + 1A41E SB%EXC(0004A) Type=0.0 Nibs=5
EOLSCN =  08AA7 SG%EXC        - 075FC JP%SYS(006CD) Type=1.1 Nibs=4 Dist=014AB
EOLSN5 =  08AB1 SG%EXC        - 06E74 SG%SYS(00AD4) Type=1.1 Nibs=4 Dist=01C3D
EOLSN7 =  08AB9 SG%EXC        - 17423 FH%TFM(00975) Type=0.1 Nibs=5
EOLSTR =  2F95B SB%RAM        -
EOLXC* =  052EC SG%LDC        -
EOLXCK =  05405 SG%LDC        - 0A106 SG%FXQ(00590) Type=1.0 Nibs=4 Dist=04D01
                              + 18E2D SC%SUB(000BB) Type=0.1 Nibs=5
                              + 1C116 MB%USG(00672) Type=0.1 Nibs=5
EPS    =  0B7A1 AB%FCN        - 1E99B SB%TAB(00464) Type=1.2 Nibs=5 Dist=131FA
ERR#   =  2F7E4 SB%RAM        - 094F9 TI%ERD(00166) Type=0.0 Nibs=4
                              + 099A3 TI%ERD(00610) Type=0.0 Nibs=5
                              + 0BE90 AB%FCN(00857) Type=0.0 Nibs=5 Offset=      2
ERR01  =  02E2B JP%PR2        - 02A01 JP%PR1(00461) Type=1.0 Nibs=3 Dist=0042A
                              + 02B91 JP%PR1(005F1) Type=1.0 Nibs=3 Dist=0029A
                              + 03488 JP%PR3(000B5) Type=1.0 Nibs=3 Dist=00660
ERR02  =  02E35 JP%PR2        - 036B6 JP%PR3(002E0) Type=1.0 Nibs=4 Dist=00881
ERR03  =  02E3F JP%PR2        - 0380C JP%PR3(00436) Type=1.0 Nibs=4 Dist=009CD
ERR04  =  02E5C JP%PR2        - 039FB JP%PR3(00625) Type=1.0 Nibs=4 Dist=00B9F
ERR05  =  02E66 JP%PR2        - 02B09 JP%PR1(00569) Type=1.0 Nibs=3 Dist=0035D
                              + 034B6 JP%PR3(000E0) Type=1.0 Nibs=3 Dist=00650
ERR06  =  02E70 JP%PR2        - 02B50 JP%PR1(005B0) Type=1.0 Nibs=3 Dist=00320
ERR08  =  02E81 JP%PR2        - 02854 JP%PR1(002B4) Type=1.0 Nibs=3 Dist=0062D
                              + 03C13 JP%PR3(0083D) Type=1.0 Nibs=4 Dist=00D92
ERR09  =  02E8B JP%PR2        - 03705 JP%PR3(0032F) Type=1.0 Nibs=4 Dist=0087A
ERR10  =  02E95 JP%PR2        - 0368C JP%PR3(002B6) Type=1.0 Nibs=4 Dist=007F7
ERR11  =  02F02 JP%PR2        - 03ABB JP%PR3(006E5) Type=1.0 Nibs=4 Dist=00BB9
ERR3   =  02E52 JP%PR2        -
ERRADR =  2F688 SB%RAM        - 07FA1 JP%EXC(001C5) Type=0.0 Nibs=5
ERRDSP =  02F3B JP%PR2        - 02788 JP%PR1(001E8) Type=1.0 Nibs=3 Dist=007B3
ERRL   =  0BED0 AB%FCN        - 1E9BF SB%TAB(00488) Type=1.2 Nibs=5 Dist=12AEF
ERRL#  =  2F7EC SB%RAM        - 0BED2 AB%FCN(00899) Type=0.0 Nibs=5
ERRLCH =  2F97C SB%RAM        -
ERRM$  =  097FC TI%ERD        - 1DE8F JP%TAB(0018B) Type=1.2 Nibs=5 Dist=14693
ERRM$f =  09806 TI%ERD        -
ERRMSG =  1593B AB%CLC        - 16780 AB%ED:(0010E) Type=1.1 Nibs=4 Dist=00E45
ERRMST =  1F3E0 TI%ERM        - 0955C TI%ERD(001C9) Type=0.0 Nibs=5
ERRN   =  0BE8E AB%FCN        - 1E9C8 SB%TAB(00491) Type=1.2 Nibs=5 Dist=12B3A
ERRRTN =  074ED JP%SYS        - 093A9 TI%ERD(00016) Type=1.0 Nibs=4 Dist=01EBC
```

```
ERRSUB =  2F683 SB%RAM        - 07A47 JP%SYS(00B18) Type=0.0 Nibs=5
                              + 07B2A JP%SYS(00BFB) Type=0.0 Nibs=5
                              + 07E42 JP%EXC(00066) Type=0.0 Nibs=5
                              + 0903E SG%EXC(009F6) Type=0.0 Nibs=5
                              + 0952A TI%ERD(00197) Type=0.0 Nibs=4
                              + 19512 SC%SUB(007A0) Type=0.0 Nibs=5
ERRX0  =  02F08 JP%PR2        - 03E8D JP%PR3(00AB7) Type=1.0 Nibs=4 Dist=00F85
ERRX10 =  02F11 JP%PR2        -
ERRX20 =  02F50 JP%PR2        -
ESCSEQ =  023C1 SB%DSP        - 096E3 TI%ERD(00350) Type=1.0 Nibs=4 Dist=07322
                              + 15270 MN%ED:(005F0) Type=0.1 Nibs=5
ESCSTA =  2F47B SB%RAM        - 01C79 SB%DSP(003E9) Type=0.0 Nibs=5
                              + 01E3F SB%DSP(005AF) Type=0.0 Nibs=5
                              + 01FE4 SB%DSP(00754) Type=0.0 Nibs=5
                              + 1022E MN%CNF(0009C) Type=0.0 Nibs=5
                              + 104A5 MN%CNF(00313) Type=0.0 Nibs=5
                              + 10E76 MN%CNF(00CE4) Type=0.0 Nibs=5
                              + 14DE7 MN%ED:(00167) Type=0.0 Nibs=5
                              + 14DFB MN%ED:(0017B) Type=0.0 Nibs=5
ETMRDC =  0554D SG%LDC        -
EX-112 =  0CF44 JT%MTH        - 0BF0B AB%FCN(008D2) Type=1.1 Nibs=4 Dist=01039
EX-115 =  0CF48 JT%MTH        -
EX-STC =  1885D SB%IO:        - 17D09 SB%RD:(00142) Type=1.0 Nibs=4 Dist=00B54
EX12   =  0D5C6 SM%MTH        - 0B716 AB%FCN(000DD) Type=1.1 Nibs=4 Dist=01EB0
EX15M  =  0D5CA SM%MTH        - 0D1C3 JT%MTH(00E9C) Type=1.0 Nibs=3 Dist=00407
                              + 0D227 JT%MTH(00F00) Type=1.1 Nibs=3 Dist=003A3
EX15S  =  0D5CE SM%MTH        -
EXAB1  =  0D3E7 JT%MTH        - 0BF11 AB%FCN(008D8) Type=1.1 Nibs=4 Dist=014D6
                              + 0DCA3 SM%MTH(0086E) Type=1.1 Nibs=4 Dist=008BC
EXAB2  =  0D40E JT%MTH        -
EXACT  =  128B0 MN%TM:        -
EXACTT =  12DD6 MN%TM:        - 1DEA1 JP%TAB(0019D) Type=1.2 Nibs=5 Dist=0B0CB
EXCAD+ =  08631 JP%EXC        -
EXCADR =  0862A JP%EXC        - 074AD JP%SYS(0057E) Type=1.1 Nibs=4 Dist=0117D
EXCHRe =  02E81 JP%PR2        -
EXCPAR =  187E8 SB%IO:        - 1ADC2 SB%VAL(00054) Type=1.1 Nibs=4 Dist=025DA
EXDCLP =  0592E SB%EXD        -
EXF    =  0D5DF SM%MTH        -
EXITBP =  0256C MB%ROM        - 0EC55 MN%BP:(00264) Type=0.1 Nibs=5
EXITRN =  076A7 JP%SYS        - 09145 SG%EXC(00AFD) Type=1.0 Nibs=4 Dist=01A9E
EXOR   =  0B8A5 AB%FCN        - 1EA8E SB%TAB(00557) Type=1.2 Nibs=5 Dist=131E9
EXP    =  0BEEE AB%FCN        - 1EAD6 SB%TAB(0059F) Type=1.2 Nibs=5 Dist=12BE8
EXP100 =  0CF79 JT%MTH        -
EXP12  =  0CF56 JT%MTH        - 0BEF7 AB%FCN(008BE) Type=1.1 Nibs=4 Dist=0105F
EXP15  =  0CF5A JT%MTH        -
EXP15M =  0CF5A JT%MTH        -
EXP16M =  0CF76 JT%MTH        -
EXP90  =  0CF60 JT%MTH        -
EXPCH# =  11A6D SC%FIL        - 191DD SC%SUB(0046B) Type=1.1 Nibs=4 Dist=07770
                              + 1945B SC%SUB(006E9) Type=1.1 Nibs=4 Dist=079EE
EXPER2 =  080BA JP%EXC        -
EXPEX+ =  0F182 AB%EXP        - 09B8D SG%FXQ(00017) Type=1.1 Nibs=4 Dist=055F5
                              + 09D39 SG%FXQ(001C3) Type=1.1 Nibs=4 Dist=05449
                              + 09EA0 SG%FXQ(0032A) Type=1.1 Nibs=4 Dist=052E2
                              + 0AC46 AB%REG(0004E) Type=1.1 Nibs=4 Dist=0453C
EXPEX- =  0F178 AB%EXP        - 07E78 JP%EXC(0009C) Type=1.1 Nibs=4 Dist=07300
                              + 089D8 SG%EXC(00390) Type=1.1 Nibs=4 Dist=067A0
                              + 0F5EA AB%ASN(00010) Type=1.1 Nibs=3 Dist=00472
                              + 0F5F5 AB%ASN(0001B) Type=1.1 Nibs=3 Dist=0047D
                              + 1114C SC%FIL(000F9) Type=1.1 Nibs=4 Dist=01FD4
```

```
                                + 111E7 SC%FIL(00194) Type=1.1 Nibs=4 Dist=0206F
                                + 136EC PM%FLG(002A3) Type=1.0 Nibs=4 Dist=04574
                                + 13E54 SC%DAT(0059F) Type=1.1 Nibs=4 Dist=04CDC
                                + 143D7 SC%DAT(00B22) Type=1.1 Nibs=4 Dist=0525F
                                + 1C642 SG%POK(00154) Type=0.1 Nibs=5
                                + 1DBFA TI%FX4(00050) Type=0.1 Nibs=5
EXPEX1 =  0F186 AB%EXP          -
EXPEXC =  0F186 AB%EXP          - 080A2 JP%EXC(002C6) Type=1.1 Nibs=4 Dist=070E4
                                + 0886F SG%EXC(00227) Type=1.1 Nibs=4 Dist=06917
                                + 08AFE SG%EXC(004B6) Type=1.1 Nibs=4 Dist=06688
                                + 08D94 SG%EXC(0074C) Type=1.1 Nibs=4 Dist=063F2
                                + 09183 SG%EXC(00B3B) Type=1.1 Nibs=4 Dist=06003
                                + 0E249 PM%STA(0041A) Type=1.0 Nibs=4 Dist=00F3D
                                + 0EA9D MN%BP:(000AC) Type=1.1 Nibs=3 Dist=006E9
                                + 0EFE4 MN%UTL(00376) Type=1.0 Nibs=3 Dist=001A2
                                + 12CFA MN%TM:(007B0) Type=1.0 Nibs=4 Dist=03B74
EXPEXj =  18528 SB%IO:          - 18C63 MN%LCK(0000B) Type=1.1 Nibs=3 Dist=0073B
                                + 1C311 MB%USG(0086D) Type=1.0 Nibs=4 Dist=03DE9
EXPM1  =  0BF02 AB%FCN          - 1DEAA JP%TAB(001A6) Type=1.2 Nibs=5 Dist=11FA8
EXPON  =  0B70D AB%FCN          - 1DEB3 JP%TAB(001AF) Type=1.2 Nibs=5 Dist=127A6
EXPP10 =  03FE3 SB%EXP          - 18809 SB%IO:(00A4D) Type=0.1 Nibs=5
EXPPAR =  03FD9 SB%EXP          - 03880 JP%PR3(004AA) Type=1.0 Nibs=3 Dist=00759
                                + 17C8D SB%RD:(000C6) Type=0.1 Nibs=5
EXPPLS =  03FDC SB%EXP          - 02AFC JP%PR1(0055C) Type=1.1 Nibs=4 Dist=014E0
EXPR   =  0F23C AB%EXP          - 0A9EC SB%FCN(000C1) Type=1.0 Nibs=4 Dist=04850
                                + 0B919 AB%FCN(002E0) Type=1.0 Nibs=4 Dist=03923
                                + 1115E SC%FIL(0010B) Type=1.1 Nibs=4 Dist=01F22
                                + 11430 SC%FIL(003DD) Type=1.1 Nibs=4 Dist=021F4
                                + 12C34 MN%TM:(006EA) Type=1.0 Nibs=4 Dist=039F8
                                + 16304 AB%CLC(00DE5) Type=1.0 Nibs=4 Dist=070C8
                                + 1EAA0 SB%TAB(00569) Type=1.2 Nibs=5 Dist=0F864
                                + 1EAA9 SB%TAB(00572) Type=1.2 Nibs=5 Dist=0F86D
EXPRDC =  05922 SB%EXD          - 05276 SG%LDC(00318) Type=1.1 Nibs=3 Dist=006AC
                                + 052C2 SG%LDC(00364) Type=1.1 Nibs=3 Dist=00660
                                + 05494 SG%LDC(00536) Type=1.1 Nibs=3 Dist=0048E
                                + 054FE SG%LDC(005A0) Type=1.1 Nibs=3 Dist=00424
                                + 055FA SG%LDC(0069C) Type=1.1 Nibs=3 Dist=00328
                                + 05606 SG%LDC(006A8) Type=1.1 Nibs=3 Dist=0031C
                                + 0571F SG%LDC(007C1) Type=1.0 Nibs=3 Dist=00203
                                + 0578B SG%LDC(0082D) Type=1.0 Nibs=3 Dist=00197
                                + 05875 SG%LDC(00917) Type=1.1 Nibs=3 Dist=000AD
                                + 058C4 SG%LDC(00966) Type=1.1 Nibs=3 Dist=0005E
                                + 0FB3A SC%TRC(000F3) Type=0.1 Nibs=5
EXPRj  =  0A9EA SB%FCN          - 0682E SG%SYS(0048E) Type=1.0 Nibs=4 Dist=041BC
EXPSKP =  1A9AC SC%REN          - 07F5F JP%EXC(00183) Type=0.1 Nibs=5
                                + 1464E SC%DAT(00D99) Type=1.1 Nibs=4 Dist=0635E
                                + 18787 SB%IO:(009CB) Type=1.1 Nibs=4 Dist=02225
EXTIF+ =  034E4 JP%PR3          - 02FF4 JP%PR2(003CE) Type=1.1 Nibs=3 Dist=004F0
EndBck =  1B85C MB%IMG          -
EndNum =  000E6 MB%IMG          -
Except =  0000C TI%EQU          - 00FE2 SB%KEY(002EB) Type=0.0 Nibs=1
                                + 0EC1C MN%BP:(0022B) Type=0.0 Nibs=1
                                + 0EC21 MN%BP:(00230) Type=0.0 Nibs=1
                                + 0EC2B MN%BP:(0023A) Type=0.0 Nibs=1
                                + 0EC52 MN%BP:(00261) Type=0.0 Nibs=1
                                + 17EC9 SB%IO:(0010D) Type=0.0 Nibs=1
ExpExc =  1C30F MB%USG          - 1C463 SB%GPH(0009E) Type=1.1 Nibs=3 Dist=00154
                                + 1C62A SG%POK(0013C) Type=1.1 Nibs=3 Dist=0031B
Expr   =  16302 AB%CLC          - 181DD SB%IO:(00421) Type=1.0 Nibs=4 Dist=01EDB
                                + 1A072 SC%SUB(01300) Type=1.0 Nibs=4 Dist=03D70
```

```
F#NFND =  11467 SC%FIL        - 14629 SC%DAT(00D74) Type=1.0 Nibs=4 Dist=031C2
                              + 18EA1 SC%SUB(0012F) Type=1.0 Nibs=4 Dist=07A3A
F-R0-0 =  2F89B SB%RAM        - 06834 SG%SYS(00494) Type=0.0 Nibs=5
                              + 0C2C5 AB%FCN(00C8C) Type=0.0 Nibs=5
                              + 1C58C SG%POK(0009E) Type=0.0 Nibs=5
F-R0-1 =  2F8A0 SB%RAM        - 08BF9 SG%EXC(005B1) Type=0.0 Nibs=5
                              + 19A61 SC%SUB(00CEF) Type=0.0 Nibs=5
                              + 1C57D SG%POK(0008F) Type=0.0 Nibs=5
                              + 1C598 SG%POK(000AA) Type=0.0 Nibs=5
F-R0-2 =  2F8A5 SB%RAM        - 0C248 AB%FCN(00C0F) Type=0.0 Nibs=5
                              + 19D5F SC%SUB(00FED) Type=0.0 Nibs=5
F-R0-3 =  2F8AA SB%RAM        - 19A18 SC%SUB(00CA6) Type=0.0 Nibs=5
                              + 19D8F SC%SUB(0101D) Type=0.0 Nibs=5
F-R1-0 =  2F8AB SB%RAM        - 0BA35 AB%FCN(003FC) Type=0.0 Nibs=5
                              + 0BB07 AB%FCN(004CE) Type=0.0 Nibs=5
                              + 0F7D3 AB%ASN(001F9) Type=0.0 Nibs=4
                              + 19E7A SC%SUB(01108) Type=0.0 Nibs=5
                              + 1A0BF SC%SUB(0134D) Type=0.0 Nibs=5
F-R1-1 =  2F8B0 SB%RAM        -
F-R1-2 =  2F8B5 SB%RAM        - 19AFE SC%SUB(00D8C) Type=0.0 Nibs=5
                              + 19DFD SC%SUB(0108B) Type=0.0 Nibs=5
F-R1-3 =  2F8BA SB%RAM        - 0BA25 AB%FCN(003EC) Type=0.0 Nibs=5
                              + 14778 SC%DAT(00EC3) Type=0.0 Nibs=5
                              + 19318 SC%SUB(005A6) Type=0.0 Nibs=5
FAC15S =  0E72B PM%STA        - 0E848 PM%STA(00A19) Type=1.1 Nibs=3 Dist=0011D
FACT   =  0E843 PM%STA        - 1EB8A SB%TAB(00653) Type=1.2 Nibs=5 Dist=10347
FAKE   =  0F452 AB%EXP        - 0BAD5 AB%FCN(0049C) Type=1.0 Nibs=4 Dist=0397D
FASCFD =  110C3 SC%FIL        - 03E7D JP%PR3(00AA7) Type=0.1 Nibs=5
FASCH  =  110FC SC%FIL        -
FCHLBL =  0782C JP%SYS        -
FCSTRT =  0E757 PM%STA        -
FDCH#  =  114AC SC%FIL        - 18E8C SC%SUB(0011A) Type=1.1 Nibs=4 Dist=079E0
FDCH#- =  114B3 SC%FIL        -
FDFIL# =  114F8 SC%FIL        -
FETCH  =  0739C JP%SYS        - 1ECAA SB%TAB(00773) Type=1.2 Nibs=5 Dist=1790E
FETCHP =  03C1D JP%PR3        - 07397 JP%SYS(00468) Type=1.2 Nibs=5 Dist=0377A
FFIB#  =  122EF SC%FIL        - 1721F FH%TFM(00771) Type=1.1 Nibs=4 Dist=04F30
FGTBL  =  00C9B SB%FGT        - 00E4D SB%KEY(00156) Type=0.0 Nibs=5
FIB#RS =  1713D FH%TFM        -
FIB#SV =  17134 FH%TFM        -
FIBAD- =  11478 SC%FIL        -
FIBADR =  11457 SC%FIL        - 13E1D SC%DAT(00568) Type=1.1 Nibs=4 Dist=029C6
FIBOFF =  12132 SC%FIL        - 0057D SB%DVR(0057D) Type=0.1 Nibs=5
FIBON  =  1212B SC%FIL        - 10EDB MN%CNF(00D49) Type=1.1 Nibs=4 Dist=01250
FILCRD =  1C879 MN%CD:        - 082B5 JP%EXC(004D9) Type=0.1 Nibs=5
FILDC* =  05759 SG%LDC        -
FILEF  =  09FB0 SG%FXQ        - 08CC4 SG%EXC(0067C) Type=1.1 Nibs=4 Dist=012EC
                              + 16D2E FH%TFM(00280) Type=0.1 Nibs=5
                              + 1D3E2 MN%CD:(00B69) Type=0.1 Nibs=5
FILEP  =  03E9C JP%PR3        -
FILEP! =  03F0F JP%PR3        - 09BE3 SG%FXQ(0006D) Type=1.1 Nibs=4 Dist=05CD4
                              + 110CA SC%FIL(00077) Type=0.1 Nibs=5
FILEP+ =  03F07 JP%PR3        - 026F3 JP%PR1(00153) Type=1.1 Nibs=4 Dist=01814
                              + 09CD4 SG%FXQ(0015E) Type=1.1 Nibs=4 Dist=05DCD
FILEP- =  03F00 JP%PR3        -
FILEP1 =  03EFC JP%PR3        -
FILFIL =  011CE TI%UTL        - 0826B JP%EXC(0048F) Type=1.1 Nibs=4 Dist=0709D
                              + 16BAF FH%TFM(00101) Type=0.1 Nibs=5
FILFMF =  09FB3 SG%FXQ        -
```

```
FILSK+ =  06F1D SG%SYS       - 09FD8 SG%FXQ(00462) Type=1.1 Nibs=4 Dist=030BB
                             + 170DC FH%TFM(0062E) Type=0.1 Nibs=5
                             + 173EC FH%TFM(0093E) Type=0.1 Nibs=5
FILSKP =  06F1B SG%SYS       - 0A19B SG%FXQ(00625) Type=1.1 Nibs=4 Dist=03280
                             + 1C7EB SG%POK(002FD) Type=0.1 Nibs=5
FILXQ$ =  09B95 SG%FXQ       - 06685 SG%SYS(002E5) Type=1.1 Nibs=4 Dist=03510
                             + 1C807 SG%POK(00319) Type=0.1 Nibs=5
FILXQ^ =  09B76 SG%FXQ       - 0733D JP%SYS(0040E) Type=1.1 Nibs=4 Dist=02839
                             + 18DC5 SC%SUB(00053) Type=0.1 Nibs=5
FINAL  =  1632C AB%BLD       - 158E2 AB%CLC(003C3) Type=1.1 Nibs=4 Dist=00A4A
FIND   =  0F563 AB%EXP       - 0B492 AB%REG(0089A) Type=1.1 Nibs=4 Dist=040D1
FINDA  =  023E3 SB%DSP       - 01070 SB%KEY(00379) Type=1.1 Nibs=4 Dist=01373
                             + 034BB JP%PR3(000E5) Type=1.1 Nibs=4 Dist=010D8
                             + 05866 SG%LDC(00908) Type=1.0 Nibs=4 Dist=03483
                             + 064E6 SG%SYS(00146) Type=1.1 Nibs=4 Dist=04103
                             + 09D71 SG%FXQ(001FB) Type=1.1 Nibs=4 Dist=0798E
                             + 1BA99 MB%IMG(00653) Type=0.1 Nibs=5
FINDAJ =  1BA97 MB%IMG       - 14FEE MN%ED:(0036E) Type=1.1 Nibs=4 Dist=06AA9
                             + 18727 SB%IO:(0096B) Type=1.1 Nibs=4 Dist=03370
                             + 1A794 SC%REN(00218) Type=1.1 Nibs=4 Dist=01303
                             + 1AA59 SC%REN(004DD) Type=1.1 Nibs=4 Dist=0103E
                             + 1D069 MN%CD:(007F0) Type=1.1 Nibs=4 Dist=015D2
FINDDO =  023E0 SB%DSP       - 17F5D SB%IO:(001A1) Type=0.1 Nibs=5
FINDF  =  09F77 SG%FXQ       - 07062 JP%SYS(00133) Type=1.1 Nibs=4 Dist=02F15
                             + 0844D JP%EXC(00671) Type=1.0 Nibs=4 Dist=01B2A
                             + 11B34 SC%FIL(00AE1) Type=1.0 Nibs=4 Dist=07BBD
FINDF+ =  09F63 SG%FXQ       - 065A7 SG%SYS(00207) Type=1.1 Nibs=4 Dist=039BC
                             + 06AAC SG%SYS(0070C) Type=1.1 Nibs=4 Dist=034B7
                             + 06CAF SG%SYS(0090F) Type=1.1 Nibs=4 Dist=032B4
                             + 190FC SC%SUB(0038A) Type=0.1 Nibs=5
                             + 1C815 SG%POK(00327) Type=0.1 Nibs=5
FINDFS =  08448 JP%EXC       - 16DA7 FH%TFM(002F9) Type=0.1 Nibs=5
FINDL  =  0FFE4 JP%MEM       - 1A60B SC%REN(0008F) Type=0.1 Nibs=5
                             + 1A63B SC%REN(000BF) Type=0.1 Nibs=5
FINDL+ =  0FFE6 JP%MEM       - 069F2 SG%SYS(00652) Type=0.1 Nibs=5
FINDLO =  0FFFD JP%MEM       - 06B5D SG%SYS(007BD) Type=0.1 Nibs=5
FINDLB =  07786 JP%SYS       -
FINDLR =  0FFDE JP%MEM       - 069D6 SG%SYS(00636) Type=0.1 Nibs=5
                             + 07151 JP%SYS(00222) Type=0.1 Nibs=5
                             + 073D9 JP%SYS(004AA) Type=0.1 Nibs=5
FINDWF =  09FA9 SG%FXQ       - 17A6C MN%FRP(00039) Type=0.1 Nibs=5
FINITA =  0CD03 JT%MTH       - 0D702 SM%MTH(002CD) Type=1.0 Nibs=4 Dist=009FF
                             + 0E928 PM%STA(00AF9) Type=1.0 Nibs=4 Dist=01C25
FINITC =  0CD0F JT%MTH       - 0D4DE SM%MTH(000A9) Type=1.1 Nibs=3 Dist=007CF
                             + 0D57E SM%MTH(00149) Type=1.1 Nibs=4 Dist=0086F
FINLIN =  18A3A SB%IO:       - 004F6 SB%DVR(004F6) Type=0.1 Nibs=5
                             + 18CF5 MN%LCK(0009D) Type=1.1 Nibs=3 Dist=002BB
                             + 18D4B MN%LCK(000F3) Type=1.1 Nibs=3 Dist=00311
FIRSTC =  2F47C SB%RAM       - 018C1 SB%DSP(00031) Type=0.0 Nibs=2
                             + 01F6B SB%DSP(006DB) Type=0.0 Nibs=5
                             + 02127 SB%DSP(00897) Type=0.0 Nibs=5
                             + 023A8 SB%DSP(00B18) Type=0.0 Nibs=5
                             + 166E3 AB%ED:(00071) Type=0.0 Nibs=4
                             + 16734 AB%ED:(000C2) Type=0.0 Nibs=5
                             + 16748 AB%ED:(000D6) Type=0.0 Nibs=5
                             + 16A42 AB%ED:(003D0) Type=0.0 Nibs=5
FIXDC  =  05493 SG%LDC       - 092B7 SG%EXC(00C6F) Type=1.2 Nibs=5 Dist=03E24
                             + 0EF99 MN%UTL(0032B) Type=1.2 Nibs=5 Dist=09B06
                             + 0EFAF MN%UTL(00341) Type=1.2 Nibs=5 Dist=09B1C
                             + 0F002 MN%UTL(00394) Type=1.2 Nibs=5 Dist=09B6F
```

```
                                + 0F069 MN%UTL(003FB) Type=1.2 Nibs=5 Dist=09BD6
                                + 12C90 MN%TM:(00746) Type=1.2 Nibs=5 Dist=0D7FD
                                + 12CFE MN%TM:(007B4) Type=1.2 Nibs=5 Dist=0D86B
                                + 12D22 MN%TM:(007D8) Type=1.2 Nibs=5 Dist=0D88F
                                + 12D92 MN%TM:(00848) Type=1.2 Nibs=5 Dist=0D8FF
                                + 1C61F SG%POK(00131) Type=1.2 Nibs=5 Dist=1718C
FIXP   = 02A6E JP%PR1           - 02D54 JP%PR2(0012E) Type=1.2 Nibs=3 Dist=002E6
                                + 03385 JP%PR2(0075F) Type=1.0 Nibs=4 Dist=00917
                                + 0EF9E MN%UTL(00330) Type=1.2 Nibs=5 Dist=0C530
                                + 0EFB4 MN%UTL(00346) Type=1.2 Nibs=5 Dist=0C546
                                + 0F007 MN%UTL(00399) Type=1.2 Nibs=5 Dist=0C599
                                + 0F06E MN%UTL(00400) Type=1.2 Nibs=5 Dist=0C600
FIXSPC = 00000 Define           - 0F7ED AB%ASN(00213) Type=0.0 Nibs=1
                                + 14279 SC%DAT(009C4) Type=0.0 Nibs=1
                                + 15AAB AB%CLC(0058C) Type=0.0 Nibs=3
                                + 16B80 FH%TFM(000D2) Type=0.0 Nibs=1
                                + 174D5 FH%TFM(00A27) Type=0.0 Nibs=1
                                + 17677 FH%TFM(00BC9) Type=0.0 Nibs=1
FIXp   = 03383 JP%PR2           - 0344D JP%PR3(00077) Type=1.2 Nibs=3 Dist=000CA
                                + 03A86 JP%PR3(006B0) Type=1.0 Nibs=3 Dist=00703
FLADDR = 0126B TI%UTL           - 0137E TI%UTL(0024D) Type=1.1 Nibs=3 Dist=00113
                                + 06D1D SG%SYS(0097D) Type=1.1 Nibs=4 Dist=05AB2
FLAG   = 13517 PM%FLG           - 1DEC5 JP%TAB(001C1) Type=1.2 Nibs=5 Dist=0A9AE
FLDE90 = 011C6 TI%UTL           - 08545 JP%EXC(00769) Type=1.0 Nibs=4 Dist=0737F
FLDEV+ = 01151 TI%UTL           - 08338 JP%EXC(0055C) Type=1.1 Nibs=4 Dist=071E7
FLDEVX = 01154 TI%UTL           - 16B26 FH%TFM(00078) Type=0.1 Nibs=5
                                + 16C4E FH%TFM(001A0) Type=0.1 Nibs=5
FLGREG = 2F6E9 SB%RAM           - 0BDB8 AB%FCN(0077F) Type=0.0 Nibs=5 Offset=    -14
                                + 134E8 PM%FLG(0009F) Type=0.0 Nibs=5
                                + 13575 PM%FLG(0012C) Type=0.0 Nibs=5 Offset=     -2
                                + 135B8 PM%FLG(0016F) Type=0.0 Nibs=4 Offset=    -14
                                + 1366C PM%FLG(00223) Type=0.0 Nibs=5 Offset=     -1
FLIP   = 07DE6 JP%EXC           - 1DF1F JP%TAB(0021B) Type=1.2 Nibs=5 Dist=16139
FLIP10 = 0DB9C SM%MTH           -
FLIP11 = 0DBAB SM%MTH           -
FLIP8  = 0DB8D SM%MTH           -
FLIPDC = 0584C SG%LDC           - 07DDC JP%EXC(00000) Type=1.2 Nibs=5 Dist=02590
FLIPP  = 03D9A JP%PR3           - 07DE1 JP%EXC(00005) Type=1.2 Nibs=5 Dist=04047
FLOAT  = 1B322 AB%UTL           - 12A57 MN%TM:(0050D) Type=0.1 Nibs=5
FLOATA = 12A55 MN%TM:           - 0BEE3 AB%FCN(008AA) Type=1.1 Nibs=4 Dist=06B72
FLOOR  = 0E862 PM%STA           -
FLSHIO = 12207 SC%FIL           -
FLSKPB = 06F19 SG%SYS           - 0A0E3 SG%FXQ(0056D) Type=1.1 Nibs=4 Dist=031CA
                                + 1C7A0 SG%POK(002B2) Type=0.1 Nibs=5
FLTDH  = 1B223 AB%UTL           - 0BDAC AB%FCN(00773) Type=0.1 Nibs=5
                                + 18006 SB%IO:(0024A) Type=1.1 Nibs=4 Dist=0321D
                                + 19AA7 SC%SUB(00D35) Type=1.1 Nibs=4 Dist=0177C
                                + 1BFE4 MB%USG(00540) Type=1.1 Nibs=4 Dist=00DC1
                                + 1C4FE SG%POK(00010) Type=1.1 Nibs=4 Dist=012DB
FLTYPp = 03E71 JP%PR3           -
FLUSHA = 121E0 SC%FIL           - 0760E JP%SYS(006DF) Type=0.1 Nibs=5
FN     = 199BE SC%SUB           - 04B7C AB%LEX(00296) Type=0.0 Nibs=5
                                + 1E9FE SB%TAB(004C7) Type=1.2 Nibs=5 Dist=05040
FN-GO  = 04B31 AB%LEX           - 1E5A2 SB%TAB(0006B) Type=1.2 Nibs=5 Dist=19A71
FNCK   = 051DC SG%LDC           - 0FAFA SC%TRC(000B3) Type=0.1 Nibs=5
FNDC   = 05219 SG%LDC           -
FNDCLR = 1DAEF MN%CD:           -
FNDDO+ = 023DD SB%DSP           - 07C88 JP%SYS(00D59) Type=1.1 Nibs=4 Dist=058AB
                                + 091E5 SG%EXC(00B9D) Type=1.1 Nibs=4 Dist=06E08
FNDFBF = 14602 SC%DAT           - 112E0 SC%FIL(0028D) Type=1.1 Nibs=4 Dist=03322
```

```
FNDFCN =  1A0A1 SC%SUB        - 16025 AB%CLC(00B06) Type=1.1 Nibs=4 Dist=0407C
FNDMK- =  122E8 SC%FIL        - 191EA SC%SUB(00478) Type=1.1 Nibs=4 Dist=06F02
FNDMRK =  122D1 SC%FIL        -
FNMISS =  1999D SC%SUB        -
FNP    =  03138 JP%PR2        -
FNP+   =  03138 JP%PR2        - 02AEE JP%PR1(0054E) Type=1.0 Nibs=3 Dist=0064A
FNPWDS =  0D3C0 JT%MTH        - 0E72D PM%STA(008FE) Type=1.1 Nibs=4 Dist=0136D
FNRTN1 =  0F216 AB%EXP        - 0B7BB AB%FCN(00182) Type=1.0 Nibs=4 Dist=03A5B
                              + 0E727 PM%STA(008F8) Type=1.0 Nibs=4 Dist=00AEF
                              + 12BCE MN%TM:(00684) Type=1.0 Nibs=4 Dist=039B8
                              + 138B1 PM%FLG(00468) Type=1.0 Nibs=4 Dist=0469B
FNRTN2 =  0F219 AB%EXP        - 09282 SG%EXC(00C3A) Type=1.0 Nibs=4 Dist=05F97
                              + 0BEC3 AB%FCN(0088A) Type=1.0 Nibs=4 Dist=03356
FNRTN3 =  0F235 AB%EXP        - 0BB39 AB%FCN(00500) Type=1.0 Nibs=4 Dist=036FC
FNRTN4 =  0F238 AB%EXP        - 0BDE2 AB%FCN(007A9) Type=1.0 Nibs=4 Dist=03456
                              + 0E3DF PM%STA(005B0) Type=1.0 Nibs=4 Dist=00E59
                              + 1376C PM%FLG(00323) Type=1.0 Nibs=4 Dist=04534
FOR    =  0865A SG%EXC        - 1EC7D SB%TAB(00746) Type=1.2 Nibs=5 Dist=16623
FORDC  =  052AA SG%LDC        - 08650 SG%EXC(00008) Type=1.2 Nibs=5 Dist=033A6
FORP   =  0341D JP%PR3        - 08655 SG%EXC(0000D) Type=1.2 Nibs=5 Dist=05238
FORSTK =  2F59E SB%RAM        - 074F1 JP%SYS(005C2) Type=0.0 Nibs=5
                              + 08889 SG%EXC(00241) Type=0.0 Nibs=4
                              + 0899A SG%EXC(00352) Type=0.0 Nibs=5
                              + 091FD SG%EXC(00BB5) Type=0.0 Nibs=5
                              + 0A6B0 SG%FXQ(00B3A) Type=0.0 Nibs=5
                              + 0AC70 AB%REG(00078) Type=0.0 Nibs=5
                              + 0B21C AB%REG(00624) Type=0.0 Nibs=5
                              + 0EE78 MN%UTL(0020A) Type=0.0 Nibs=5
                              + 0FF19 SC%TRC(004D2) Type=0.0 Nibs=5
                              + 134C9 PM%FLG(00080) Type=0.0 Nibs=5
                              + 13D4E SC%DAT(00499) Type=0.0 Nibs=5
                              + 13E34 SC%DAT(0057F) Type=0.0 Nibs=5
                              + 155C2 AB%CLC(000A3) Type=0.0 Nibs=2
                              + 1948A SC%SUB(00718) Type=0.0 Nibs=5
                              + 1955F SC%SUB(007ED) Type=0.0 Nibs=5
                              + 19741 SC%SUB(009CF) Type=0.0 Nibs=5
FORUPD =  0A6AE SG%FXQ        - 0910F SG%EXC(00AC7) Type=1.0 Nibs=4 Dist=0159F
FP     =  0E851 PM%STA        - 1E965 SB%TAB(0042E) Type=1.2 Nibs=5 Dist=10114
FPOLL  =  1250A JP%POL        - 07622 JP%SYS(006F3) Type=0.1 Nibs=5
                              + 0A990 SB%FCN(00065) Type=1.1 Nibs=4 Dist=07B7A
                              + 0BC5C AB%FCN(00623) Type=1.1 Nibs=4 Dist=068AE
                              + 10072 JP%MEM(00113) Type=1.1 Nibs=4 Dist=02498
                              + 10F06 MN%CNF(00D74) Type=1.1 Nibs=4 Dist=01604
                              + 11B02 SC%FIL(00AAF) Type=1.0 Nibs=4 Dist=00A08
                              + 13AC1 SC%DAT(0020C) Type=1.0 Nibs=4 Dist=015B7
                              + 14E1A MN%ED:(0019A) Type=1.1 Nibs=4 Dist=02910
                              + 14EA6 MN%ED:(00226) Type=1.1 Nibs=4 Dist=0299C
                              + 17042 FH%TFM(00594) Type=1.1 Nibs=4 Dist=04B38
                              + 19DD6 SC%SUB(01064) Type=1.1 Nibs=4 Dist=078CC
                              + 19EBC SC%SUB(0114A) Type=1.1 Nibs=4 Dist=079B2
FPOLLj =  1B997 MB%IMG        - 1BAD7 MB%USG(00033) Type=1.1 Nibs=3 Dist=00140
                              + 1CF2D MN%CD:(006B4) Type=1.0 Nibs=4 Dist=01596
FPOLLx =  1B993 MB%IMG        - 1BB90 MB%USG(000EC) Type=1.1 Nibs=3 Dist=001FD
                              + 1BD02 MB%USG(0025E) Type=1.1 Nibs=3 Dist=0036F
FPoll  =  07620 JP%SYS        - 00424 SB%DVR(00424) Type=1.0 Nibs=4 Dist=071FC
                              + 02636 JP%PR1(00096) Type=1.1 Nibs=4 Dist=04FEA
                              + 09010 SG%EXC(009C8) Type=1.1 Nibs=4 Dist=019F0
                              + 0946D TI%ERD(000DA) Type=1.1 Nibs=4 Dist=01E4D
FRAC15 =  0C70E JT%MTH        - 09EFF SG%FXQ(00389) Type=1.1 Nibs=4 Dist=0280F
                              + 0E857 PM%STA(00A28) Type=1.1 Nibs=4 Dist=02149
```

```
FRC15F =  0C721 JT%MTH        -
FREE07 =  033BD JP%PR2        - 03464 JP%PR3(0008E) Type=1.0 Nibs=3 Dist=000A7
FREE15 =  033B3 JP%PR2        - 03D87 JP%PR3(009B1) Type=1.0 Nibs=4 Dist=009D4
FREEp  =  0338C JP%PR2        - 17A33 MN%FRP(00000) Type=1.2 Nibs=5 Dist=146A7
FROMDT =  1323B MN%TM:        - 1C92B MN%CD:(000B2) Type=0.1 Nibs=5
FRPORT =  17A38 MN%FRP        - 1DEE0 JP%TAB(001DC) Type=1.2 Nibs=5 Dist=064A8
FRange =  0B46A SB%EXP        - 05C64 SB%EXD(00342) Type=0.0 Nibs=4
FSPC10 =  03CE9 JP%PR3        - 02A4E JP%PR1(004AE) Type=1.0 Nibs=4 Dist=0129B
FSPC12 =  03CF4 JP%PR3        - 02737 JP%PR1(00197) Type=1.1 Nibs=4 Dist=015BD
FSPCER =  09DFE SG%FXQ        - 06FA9 JP%SYS(0007A) Type=1.0 Nibs=4 Dist=02E55
FSPECe =  02F02 JP%PR2        -
FSPECj =  06C92 SG%SYS        - 06F7E JP%SYS(0004F) Type=1.1 Nibs=3 Dist=002EC
FSPECp =  03CC5 JP%PR3        -
FSPECx =  09F2D SG%FXQ        - 06C94 SG%SYS(008F4) Type=1.0 Nibs=4 Dist=03299
                              + 08239 JP%EXC(0045D) Type=1.1 Nibs=4 Dist=01CF4
                              + 117E2 SC%FIL(0078F) Type=1.0 Nibs=4 Dist=078B5
FTBSCH =  11093 SC%FIL        -
FTCHKY =  1AAA1 SG%KEY        - 073AD JP%SYS(0047E) Type=0.1 Nibs=5
FTYPD+ =  068F7 SG%SYS        - 05650 SG%LDC(006F2) Type=1.1 Nibs=4 Dist=012A7
FTYPDC =  06902 SG%SYS        -
FTYPE  =  1FF2E SC%TAB        - 11061 SC%FIL(0000E) Type=0.0 Nibs=5
                              + 110D7 SC%FIL(00084) Type=0.0 Nibs=5
FTYPER =  115B5 SC%FIL        - 1007C JP%MEM(0011D) Type=1.0 Nibs=4 Dist=01539
                              + 142BE SC%DAT(00A09) Type=1.0 Nibs=4 Dist=02D09
FTYPF# =  11059 SC%FIL        - 1C8AB MN%CD:(00032) Type=0.1 Nibs=5
                              + 1DA8C MN%CD:(01213) Type=0.1 Nibs=5
FTYPFD =  11053 SC%FIL        - 06904 SG%SYS(00564) Type=0.1 Nibs=5
                              + 0A3E1 SG%FXQ(0086B) Type=1.1 Nibs=4 Dist=06C72
FUNCD0 =  2F8BB SB%RAM        - 04A4E AB%LEX(00168) Type=0.0 Nibs=5
                              + 04BF6 AB%LEX(00310) Type=0.0 Nibs=5
                              + 0BC3E AB%FCN(00605) Type=0.0 Nibs=5
                              + 0BC66 AB%FCN(0062D) Type=0.0 Nibs=5
FUNCD1 =  2F8C0 SB%RAM        - 1ADAF SB%VAL(00041) Type=0.0 Nibs=5
                              + 1AE43 SB%VAL(000D5) Type=0.0 Nibs=5
FUNCR0 =  2F89B SB%RAM        - 14ECA MN%ED:(0024A) Type=0.0 Nibs=5 Offset=     -2
                              + 14EE8 MN%ED:(00268) Type=0.0 Nibs=5 Offset=     -2
                              + 14F16 MN%ED:(00296) Type=0.0 Nibs=5 Offset=     -2
                              + 15142 MN%ED:(004C2) Type=0.0 Nibs=5 Offset=     -2
                              + 15FA8 AB%CLC(00A89) Type=0.0 Nibs=5
                              + 15FF2 AB%CLC(00AD3) Type=0.0 Nibs=5
                              + 16032 AB%CLC(00B13) Type=0.0 Nibs=5
                              + 18051 SB%IO:(00295) Type=0.0 Nibs=5
                              + 1805C SB%IO:(002A0) Type=0.0 Nibs=5
FUNCR1 =  2F8AB SB%RAM        -
FUNNY  =  04C72 AB%LEX        - 04B57 AB%LEX(00271) Type=0.0 Nibs=5

GDISP  =  1C462 SB%GPH        - 1DEE9 JP%TAB(001E5) Type=1.2 Nibs=5 Dist=01A87
GDISP$ =  1C3C7 SB%GPH        - 1DEF2 JP%TAB(001EE) Type=1.2 Nibs=5 Dist=01B2B
GETARG =  1113E SC%FIL        -
GETAVM =  1864D SB%IO:        -
GETBUF =  12247 SC%FIL        -
GETC#- =  1142E SC%FIL        - 18E70 SC%SUB(000FE) Type=1.1 Nibs=4 Dist=07A42
GETCH# =  11427 SC%FIL        - 13EC0 SC%DAT(0060B) Type=1.1 Nibs=4 Dist=02A99
                              + 14296 SC%DAT(009E1) Type=1.1 Nibs=4 Dist=02E6F
GETCNT =  1A076 SC%SUB        - 0B370 AB%REG(00778) Type=0.1 Nibs=5
GETCON =  0DAA3 SM%MTH        - 0C19A AB%FCN(00B61) Type=1.1 Nibs=4 Dist=01909
GETDIM =  0AD6B AB%REG        -
GETLEE =  017D7 SB%CMD        - 028E0 JP%PR1(00340) Type=1.1 Nibs=4 Dist=01109
GETMSK =  01BBA SB%DSP        -
GETNAM =  1A085 SC%SUB        -
```

```
GETPR  =  06BF8 SG%SYS        - 08D6A SG%EXC(00722) Type=1.1 Nibs=4 Dist=02172
GETPR+ =  06BF5 SG%SYS        -
GETPR1 =  06BFB SG%SYS        - 082E4 JP%EXC(00508) Type=1.1 Nibs=4 Dist=016E9
                              + 0A0D8 SG%FXQ(00562) Type=1.1 Nibs=4 Dist=034DD
                              + 0A152 SG%FXQ(005DC) Type=1.1 Nibs=4 Dist=03557
                              + 0A3D2 SG%FXQ(0085C) Type=1.1 Nibs=4 Dist=037D7
                              + 1C796 SG%POK(002A8) Type=0.1 Nibs=5
GETPR0 =  06BEE SG%SYS        - 0771F JP%SYS(007F0) Type=1.1 Nibs=4 Dist=00B31
                              + 07772 JP%SYS(00843) Type=1.1 Nibs=4 Dist=00B84
GETPRe =  0776C JP%SYS        -
GETPeF =  0771D JP%SYS        - 1015C JP%MEM(001FD) Type=0.1 Nibs=5
GETRE# =  111C6 SC%FIL        - 13EC6 SC%DAT(00611) Type=1.1 Nibs=4 Dist=02D00
                              + 1429C SC%DAT(009E7) Type=1.1 Nibs=4 Dist=030D6
GETRG+ =  0EE76 MN%UTL        - 0669E SG%SYS(002FE) Type=0.1 Nibs=5
                              + 0EAA1 MN%BP:(000B0) Type=1.1 Nibs=3 Dist=003D5
                              + 0EAAB MN%BP:(000BA) Type=1.1 Nibs=3 Dist=003CB
GETSA  =  0E551 PM%STA        -
GETSPC =  14A68 SC%DAT        -
GETST  =  0F5B6 AB%EXP        - 086B3 SG%EXC(0006B) Type=1.1 Nibs=4 Dist=06F03
                              + 162FB AB%CLC(00DDC) Type=1.1 Nibs=4 Dist=06D45
                              + 18DDC SC%SUB(0006A) Type=0.1 Nibs=5
GETST* =  07716 JP%SYS        - 06D09 SG%SYS(00969) Type=1.1 Nibs=4 Dist=00A0D
GETST- =  07728 JP%SYS        -
GETST1 =  0772C JP%SYS        - 06AD2 SG%SYS(00732) Type=1.1 Nibs=4 Dist=00C5A
GETSTA =  01B93 SB%DSP        - 004EB SB%DVR(004EB) Type=1.1 Nibs=4 Dist=016A8
                              + 0072A SB%DVR(0072A) Type=1.1 Nibs=4 Dist=01469
GETSTC =  07726 JP%SYS        - 18FD4 SC%SUB(00262) Type=0.1 Nibs=5
GETSTe =  0775F JP%SYS        -
GETVAL =  0DAB2 SM%MTH        -
GMEME  =  0CCF7 JT%MTH        -
GNXCR+ =  03067 JP%PR2        - 037D9 JP%PR3(00403) Type=1.1 Nibs=3 Dist=00772
                              + 037EF JP%PR3(00419) Type=1.1 Nibs=3 Dist=00788
                              + 0410C SB%EXP(00137) Type=1.1 Nibs=4 Dist=010A5
                              + 0416B SB%EXP(00196) Type=1.1 Nibs=4 Dist=01104
                              + 1606A AB%CLC(00B4B) Type=0.1 Nibs=5
GNXTCR =  03064 JP%PR2        - 0273D JP%PR1(0019D) Type=1.1 Nibs=4 Dist=00927
                              + 02B21 JP%PR1(00581) Type=1.1 Nibs=3 Dist=00543
                              + 03422 JP%PR3(0004C) Type=1.1 Nibs=3 Dist=003BE
                              + 036DF JP%PR3(00309) Type=1.1 Nibs=3 Dist=0067B
                              + 03C19 JP%PR3(00843) Type=1.0 Nibs=4 Dist=00BB5
                              + 048BD SB%EXP(008E8) Type=1.1 Nibs=4 Dist=01859
                              + 048F4 AB%LEX(0000E) Type=1.1 Nibs=4 Dist=01890
                              + 0493D AB%LEX(00057) Type=1.1 Nibs=4 Dist=018D9
                              + 04B49 AB%LEX(00263) Type=1.1 Nibs=4 Dist=01AE5
                              + 04CE6 AB%LEX(00400) Type=1.1 Nibs=4 Dist=01C82
                              + 1897D SB%IO:(00BC1) Type=0.1 Nibs=5
GOSBDC =  0552E SG%LDC        - 079DF JP%SYS(00AB0) Type=1.2 Nibs=5 Dist=024B1
GOSUB  =  079E9 JP%SYS        - 1ED5E SB%TAB(00827) Type=1.2 Nibs=5 Dist=17375
GOSUBP =  029F6 JP%PR1        - 079E4 JP%SYS(00AB5) Type=1.2 Nibs=5 Dist=04FEE
GOSUBp =  029F6 JP%PR1        -
GOTO   =  079FA JP%SYS        - 091F7 SG%EXC(00BAF) Type=1.0 Nibs=4 Dist=017FD
                              + 1ED67 SB%TAB(00830) Type=1.2 Nibs=5 Dist=1736D
GOTO+  =  07A06 JP%SYS        - 08070 JP%EXC(00294) Type=1.0 Nibs=3 Dist=0066A
GOTODC =  0552E SG%LDC        - 079F0 JP%SYS(00AC1) Type=1.2 Nibs=5 Dist=024C2
GOTOP  =  029F6 JP%PR1        - 079F5 JP%SYS(00AC6) Type=1.2 Nibs=5 Dist=04FFF
GOTOp  =  029F6 JP%PR1        -
GSBSTK =  2F5A3 SB%RAM        - 06CDF SG%SYS(0093F) Type=0.0 Nibs=5
                              + 06D8B SG%SYS(009EB) Type=0.0 Nibs=5
                              + 08F27 SG%EXC(008DF) Type=0.0 Nibs=5
                              + 08FA0 SG%EXC(00958) Type=0.0 Nibs=5
```

```
                                + 0B377 AB%REG(0077F) Type=0.0 Nibs=5
                                + 123C6 JP%POL(000AB) Type=0.0 Nibs=5
                                + 16FD3 FH%TFM(00525) Type=0.0 Nibs=5
                                + 19D00 SC%SUB(00F8E) Type=0.0 Nibs=5
                                + 1AE19 SB%VAL(000AB) Type=0.0 Nibs=4
                                + 1AE6C SB%VAL(000FE) Type=0.0 Nibs=5
GTEXT  =  05079 SG%LDC          - 164FF AB%BLD(001D3) Type=0.1 Nibs=5
GTEXT+ =  05199 SG%LDC          -
GTEXT1 =  051A5 SG%LDC          -
GTEXTM =  0508B SG%LDC          - 05CA5 SB%EXD(00383) Type=1.1 Nibs=4 Dist=00C1A
GTEXTX =  0509B SG%LDC          - 05C33 SB%EXD(00311) Type=1.1 Nibs=4 Dist=00B98
                                + 05DD7 SB%EXD(004B5) Type=1.1 Nibs=4 Dist=00D3C
GTFLAG =  1365E PM%FLG          - 126CA MN%TM:(00180) Type=1.1 Nibs=4 Dist=00F94
                                + 128BA MN%TM:(00370) Type=1.1 Nibs=4 Dist=00DA4
GTKYC+ =  08D9B SG%EXC          -
GTKYCD =  08D92 SG%EXC          - 1AAA3 SG%KEY(00002) Type=0.1 Nibs=5
GTMR#  =  080CA JP%EXC          -
GTMRA+ =  080F7 JP%EXC          - 0756E JP%SYS(0063F) Type=1.1 Nibs=4 Dist=00B89
GTMRAD =  080FF JP%EXC          -
GTPTRS =  14636 SC%DAT          -
GTPTRX =  14670 SC%DAT          - 17240 FH%TFM(00792) Type=1.1 Nibs=4 Dist=02BD0
GTSVC# =  12063 SC%FIL          -
GTXT++ =  05192 SG%LDC          -
GetEXP =  1C086 MB%USG          -

HASH1  =  1B0A1 AB%UTL          -
HASH2  =  1B0A3 AB%UTL          - 15786 AB%CLC(00267) Type=1.0 Nibs=4 Dist=0591D
                                + 16A76 AB%ED:(00404) Type=1.0 Nibs=4 Dist=0462D
HDFLT  =  1B31B AB%UTL          - 0BEC9 AB%FCN(00890) Type=0.1 Nibs=5
HEX$   =  1C85F SG%POK          - 1DE7D JP%TAB(00179) Type=1.2 Nibs=5 Dist=0161E
HEXASC =  17148 FH%TFM          - 1C552 SG%POK(00064) Type=1.1 Nibs=4 Dist=0540A
                                + 1C832 SG%POK(00344) Type=1.1 Nibs=4 Dist=056EA
HEXDEC =  0ECAF MN%UTL          - 06867 SG%SYS(004C7) Type=0.1 Nibs=5
                                + 068D9 SG%SYS(00539) Type=0.1 Nibs=5
                                + 09890 TI%ERD(004FD) Type=1.1 Nibs=4 Dist=0541F
                                + 0FF52 SC%TRC(0050B) Type=1.1 Nibs=4 Dist=012A3
                                + 1B31D AB%UTL(002FE) Type=0.1 Nibs=5
HMSSEC =  13274 MN%TM:          -
HNDL20 =  0CBDC JT%MTH          - 04706 SB%EXP(00731) Type=0.1 Nibs=5
HNDLFL =  0CBC9 JT%MTH          -
HOWARD =  04BF2 AB%LEX          -
HPSCRH =  2F97F SB%RAM          -
HTRAP  =  0CB2F JT%MTH          - 1C2B7 MB%USG(00813) Type=0.1 Nibs=5
HUGE   =  0B75D AB%FCN          -
HUGE20 =  0B760 AB%FCN          - 0CA92 JT%MTH(0076B) Type=1.0 Nibs=4 Dist=01332
HXDASC =  05FF4 SB%EXD          - 1AC84 SG%KEY(001E3) Type=0.1 Nibs=5
HXDCW  =  0ECB4 MN%UTL          - 12CEE MN%TM:(007A4) Type=1.0 Nibs=4 Dist=0403A
HXDEC  =  1C869 SG%POK          - 1DEFB JP%TAB(001F7) Type=1.2 Nibs=5 Dist=01692

I/OAL+ =  1197B SC%FIL          - 028A9 JP%PR1(00309) Type=0.1 Nibs=5
                                + 10DD7 MN%CNF(00C45) Type=1.0 Nibs=4 Dist=00BA4
                                + 10DF0 MN%CNF(00C5E) Type=1.1 Nibs=4 Dist=00B8B
                                + 1CEFE MN%CD:(00685) Type=0.1 Nibs=5
I/OALL =  1197D SC%FIL          - 002D2 SB%DVR(002D2) Type=0.1 Nibs=5
                                + 00448 SB%DVR(00448) Type=0.1 Nibs=5
                                + 0E09F PM%STA(00270) Type=1.1 Nibs=4 Dist=038DE
                                + 191B6 SC%SUB(00444) Type=1.1 Nibs=4 Dist=07839
                                + 1A51D SB%EXC(00149) Type=0.1 Nibs=5
I/OCOL =  11979 SC%FIL          - 090F3 SG%EXC(00AAB) Type=0.1 Nibs=5
                                + 10E27 MN%CNF(00C95) Type=1.1 Nibs=4 Dist=00B52
```

```
I/OCON =  11920 SC%FIL        -
I/ODAL =  11A41 SC%FIL        - 003A8 SB%DVR(003A8) Type=0.1 Nibs=5
                              + 005F2 SB%DVR(005F2) Type=0.1 Nibs=5
                              + 0E043 PM%STA(00214) Type=1.1 Nibs=4 Dist=039FE
                              + 0F117 MN%UTL(004A9) Type=1.1 Nibs=4 Dist=0292A
                              + 1CF0C MN%CD:(00693) Type=0.1 Nibs=5
I/OEX2 =  11A0F SC%FIL        - 10FB4 MN%CNF(00E22) Type=1.1 Nibs=4 Dist=00A5B
I/OEXP =  11A11 SC%FIL        -
I/OFND =  118BA SC%FIL        - 0053D SB%DVR(0053D) Type=0.1 Nibs=5
                              + 00553 SB%DVR(00553) Type=0.1 Nibs=5
                              + 097A6 TI%ERD(00413) Type=0.1 Nibs=5
                              + 1461E SC%DAT(00D69) Type=1.1 Nibs=4 Dist=02D64
                              + 191A7 SC%SUB(00435) Type=1.1 Nibs=4 Dist=078ED
I/ORES =  118FF SC%FIL        - 0F169 MN%UTL(004FB) Type=1.1 Nibs=4 Dist=02796
                              + 10F00 MN%CNF(00D6E) Type=1.1 Nibs=4 Dist=009FF
IDIV   =  0EC7B MN%UTL        - 0EB75 MN%BP:(00184) Type=1.1 Nibs=3 Dist=00106
                              + 0EBE3 MN%BP:(001F2) Type=1.1 Nibs=3 Dist=00098
                              + 0FCD1 SC%TRC(0028A) Type=1.1 Nibs=4 Dist=01056
                              + 0FCDA SC%TRC(00293) Type=1.1 Nibs=4 Dist=0105F
                              + 11863 SC%FIL(00810) Type=1.1 Nibs=4 Dist=02BE8
                              + 12CDC MN%TM:(00792) Type=1.0 Nibs=4 Dist=04061
                              + 13CD1 SC%DAT(0041C) Type=1.1 Nibs=4 Dist=05056
                              + 14BB3 SC%DAT(012FE) Type=1.1 Nibs=4 Dist=05F38
IDIV12 =  0C8F9 JT%MTH        - 0C14B AB%FCN(00B12) Type=1.1 Nibs=3 Dist=007AE
IDIV15 =  0C8FD JT%MTH        -
IDIVA  =  0EC6E MN%UTL        - 152FC MN%ED:(0067C) Type=1.1 Nibs=4 Dist=0668E
                              + 1CE54 MN%CD:(005DB) Type=0.1 Nibs=5
IF     =  09181 SG%EXC        - 1ED79 SB%TAB(00842) Type=1.2 Nibs=5 Dist=15BF8
IF12A  =  0C739 JT%MTH        - 0F9A5 AB%ASN(003CB) Type=1.1 Nibs=4 Dist=0326C
IFCK   =  03082 JP%PR2        - 03A3F JP%PR3(00669) Type=1.1 Nibs=4 Dist=009BD
IFDC   =  052C1 SG%LDC        - 09177 SG%EXC(00B2F) Type=1.2 Nibs=5 Dist=03EB6
IFP    =  03467 JP%PR3        - 0917C SG%EXC(00B34) Type=1.2 Nibs=5 Dist=05D15
ILCNTe =  02E70 JP%PR2        -
IMAGE  =  08A48 SG%EXC        - 1EE99 SB%TAB(00962) Type=1.2 Nibs=5 Dist=16451
IMDO+2 =  1BA2D MB%IMG        -
IMDO-- =  1BA24 MB%IMG        - 1BC86 MB%USG(001E2) Type=1.1 Nibs=3 Dist=00262
IMDO-2 =  1BA21 MB%IMG        -
IMGxq1 =  1BAAB MB%USG        - 1B590 MB%IMG(0014A) Type=1.0 Nibs=3 Dist=0051B
                              + 1B977 MB%IMG(00531) Type=1.0 Nibs=3 Dist=00134
IMGxqt =  1BAA4 MB%USG        - 1B901 MB%IMG(004BB) Type=1.0 Nibs=3 Dist=001A3
IMLETP =  02AF1 JP%PR1        -
IMXRTN =  1BB95 MB%USG        -
IMerr  =  1B989 MB%IMG        -
IMinit =  1B88F MB%IMG        -
IMoffs =  1BA58 MB%IMG        -
IMtbl  =  1C39D MB%USG        - 1B6DB MB%IMG(00295) Type=0.0 Nibs=5
                              + 1C1ED MB%USG(00749) Type=0.0 Nibs=5
IMxq05 =  1BADC MB%USG        -
IMxq27 =  1BB9C MB%USG        -
IN/REP =  15255 MN%ED:        -
INADDR =  2F6D4 SB%RAM        - 02DB4 JP%PR2(0018E) Type=0.0 Nibs=5
                              + 02DD3 JP%PR2(001AD) Type=0.0 Nibs=5
                              + 02F8C JP%PR2(00366) Type=0.0 Nibs=5
                              + 04F8C SG%LDC(0002E) Type=0.0 Nibs=5
                              + 057FB SG%LDC(0089D) Type=0.0 Nibs=5
INBS   =  2F6C6 SB%RAM        - 01468 TI%UTL(00337) Type=0.0 Nibs=5
                              + 01474 TI%UTL(00343) Type=0.0 Nibs=5
                              + 02628 JP%PR1(00088) Type=0.0 Nibs=5
                              + 09420 TI%ERD(0008D) Type=0.0 Nibs=5
                              + 09690 TI%ERD(002FD) Type=0.0 Nibs=5
```

```
                                    + 17781 FH%TFM(00CD3) Type=0.0 Nibs=5
                                    + 1893A SB%IO:(00B7E) Type=0.0 Nibs=4
                                    + 18A20 SB%IO:(00C64) Type=0.0 Nibs=5
INBS=C =  01466 TI%UTL              -
INF    =  0B7AA AB%FCN              - 1E992 SB%TAB(0045B) Type=1.2 Nibs=5 Dist=131E8
INF*0  =  0C607 JT%MTH              -
INFR15 =  0C73D JT%MTH              - 0E73F PM%STA(00910) Type=1.1 Nibs=4 Dist=02002
INITCL =  13121 MN%TM:              - 00245 SB%DVR(00245) Type=0.1 Nibs=5
INITMF =  117ED SC%FIL              -
INITMS =  010DC SB%KEY              - 01047 SB%KEY(00350) Type=0.0 Nibs=5
INITPT =  10D95 MN%CNF              - 01443 TI%UTL(00312) Type=0.1 Nibs=5
INPEND =  18B0D SB%IO:              -
INP010 =  18B54 SB%IO:              -
INPOFF =  18B49 SB%IO:              -
INPTDC =  05445 SG%LDC              - 18699 SB%IO:(008DD) Type=1.2 Nibs=5 Dist=13254
                                    + 18BDE SB%IO:(00E22) Type=1.2 Nibs=5 Dist=13799
INPUT  =  186A3 SB%IO:              - 1ECB3 SB%TAB(0077C) Type=1.2 Nibs=5 Dist=06610
INPUTP =  031FA JP%PR2              - 1869E SB%IO:(008E2) Type=1.2 Nibs=5 Dist=154A4
INSCUR =  00791 SB%BIT              -
INT    =  0E862 PM%STA              - 1DECE JP%TAB(001CA) Type=1.2 Nibs=5 Dist=0F66C
                                    + 1EB1E SB%TAB(005E7) Type=1.2 Nibs=5 Dist=102BC
INTA   =  2F410 SB%RAM              - 00022 SB%DVR(00022) Type=0.0 Nibs=2
                                    + 00107 SB%DVR(00107) Type=0.0 Nibs=2
INTB   =  2F420 SB%RAM              - 0002A SB%DVR(0002A) Type=0.0 Nibs=2
                                    + 000FC SB%DVR(000FC) Type=0.0 Nibs=2
INTEGR =  0AC02 AB%REG              - 1ECBC SB%TAB(00785) Type=1.2 Nibs=5 Dist=140BA
INTGR  =  0F99B AB%ASN              -
INTM   =  2F430 SB%RAM              - 000DD SB%DVR(000DD) Type=0.0 Nibs=5
INTR4  =  2F400 SB%RAM              - 00017 SB%DVR(00017) Type=0.0 Nibs=5
                                    + 0010F SB%DVR(0010F) Type=0.0 Nibs=2
                                    + 10198 MN%CNF(00006) Type=0.0 Nibs=5
INTR50 =  000DB SB%DVR              -
INTRPT =  0000F SB%DVR              -
INTSHT =  0F2A7 AB%EXP              - 0BB59 AB%FCN(00520) Type=1.0 Nibs=4 Dist=0374E
INVFSP =  066FC SG%SYS              - 09E00 SG%FXQ(0028A) Type=1.0 Nibs=4 Dist=03704
                                    + 1C7F9 SG%POK(0030B) Type=0.1 Nibs=5
INVLUT =  0B63B AB%FCN              - 1EA22 SB%TAB(004EB) Type=1.2 Nibs=5 Dist=133E7
INVNaN =  0C65F JT%MTH              - 0BC7A AB%FCN(00641) Type=1.1 Nibs=4 Dist=009E5
                                    + 0D751 SM%MTH(0031C) Type=1.0 Nibs=4 Dist=010F2
                                    + 0E52F PM%STA(00700) Type=1.1 Nibs=4 Dist=01ED0
INVRES =  0BC78 AB%FCN              - 0874F SG%EXC(00107) Type=1.1 Nibs=4 Dist=03529
INX    =  13885 PM%FLG              - 1EBE4 SB%TAB(006AD) Type=1.2 Nibs=5 Dist=0B35F
INXNIB =  2F6F9 SB%RAM              -
IOBFEN =  2F576 SB%RAM              - 119F5 SC%FIL(009A2) Type=0.0 Nibs=5
                                    + 11A8F SC%FIL(00A3C) Type=0.0 Nibs=5
IOBFST =  2F571 SB%RAM              - 0F147 MN%UTL(004D9) Type=0.0 Nibs=5
IOCNDO =  11925 SC%FIL              -
IOFNDO =  118C1 SC%FIL              - 019D6 SB%DSP(00146) Type=0.1 Nibs=5
                                    + 04A34 AB%LEX(0014E) Type=0.1 Nibs=5
                                    + 0AA0B SB%FCN(000E0) Type=1.1 Nibs=4 Dist=06EB6
                                    + 10EE8 MN%CNF(00D56) Type=1.1 Nibs=4 Dist=009D9
IOFSCR =  1188E SC%FIL              -
IP     =  0E894 PM%STA              - 1E95C SB%TAB(00425) Type=1.2 Nibs=5 Dist=100C8
IS-DSP =  2F78D SB%RAM              -
IS-INP =  2F79B SB%RAM              -
IS-PLT =  2F7A2 SB%RAM              -
IS-PRT =  2F794 SB%RAM              -
IS-TBL =  2F78D SB%RAM              - 0027D SB%DVR(0027D) Type=0.0 Nibs=5
ISRAM? =  10192 MN%CNF              - 07975 JP%SYS(00A46) Type=0.1 Nibs=5
                                    + 07C4A JP%SYS(00D1B) Type=0.1 Nibs=5
```

```
                                    + 1A319 SC%SUB(015A7) Type=0.1 Nibs=5
                                    + 1A763 SC%REN(001E7) Type=0.1 Nibs=5
IVAERR =  0E920 PM%STA              - 0EE1E MN%UTL(001B0) Type=1.0 Nibs=3 Dist=004FE
                                    + 136E6 PM%FLG(0029D) Type=1.0 Nibs=4 Dist=04DC6
IVARG  =  0D749 SM%MTH              - 0C8D5 JT%MTH(005AE) Type=1.0 Nibs=4 Dist=00E74
                                    + 0E2BF PM%STA(00490) Type=1.1 Nibs=4 Dist=00B76
IVEXPe =  02E35 JP%PR2              -
IVL    =  138A1 PM%FLG              - 1EBC0 SB%TAB(00689) Type=1.2 Nibs=5 Dist=0B31F
IVLNIB =  2F6FD SB%RAM              -
IVP    =  00004 JT%MTH              - 0D46A SM%MTH(00035) Type=0.0 Nibs=1
                                    + 0E1C5 PM%STA(00396) Type=0.0 Nibs=1
IVPARe =  02E3F JP%PR2              -
IVVARe =  02E66 JP%PR2              -
InhEOL =  00004 MB%IMG              -
Insert =  00007 SB%DSP              -
InvArg =  1929E SC%SUB              - 1A3D6 SB%EXC(00002) Type=1.0 Nibs=4 Dist=01138
                                    + 1A67A SC%REN(000FE) Type=1.0 Nibs=4 Dist=013DC
                                    + 1AD7B SB%VAL(0000D) Type=1.0 Nibs=4 Dist=01ADD
                                    + 1BFF2 MB%USG(0054E) Type=1.0 Nibs=4 Dist=02D54
                                    + 1C61B SG%POK(0012D) Type=1.0 Nibs=4 Dist=0337D
Invarg =  12D48 MN%TM:              - 192A0 SC%SUB(0052E) Type=1.0 Nibs=4 Dist=06558
IvUSGj =  1B4E8 MB%IMG              - 1C0EC MB%USG(00648) Type=1.0 Nibs=4 Dist=00C04

JPSYS  =  06F2F JP%SYS              -

KBRTCK =  09113 SG%EXC              - 1962A SC%SUB(008B8) Type=0.1 Nibs=5
                                    + 1A040 SC%SUB(012CE) Type=0.1 Nibs=5
KCOL0  =  2F46F SB%RAM              - 15304 MN%ED:(00684) Type=0.0 Nibs=5
KCOL1  =  2F46E SB%RAM              -
KCOL2  =  2F46D SB%RAM              -
KCOL3  =  2F46C SB%RAM              -
KCOL4  =  2F46B SB%RAM              -
KCOL5  =  2F46A SB%RAM              -
KCOL6  =  2F469 SB%RAM              -
KCOL7  =  2F468 SB%RAM              -
KCOL8  =  2F467 SB%RAM              -
KCOL9  =  2F466 SB%RAM              -
KCOLA  =  2F465 SB%RAM              -
KCOLB  =  2F464 SB%RAM              -
KCOLC  =  2F463 SB%RAM              -
KCOLD  =  2F462 SB%RAM              -
KEY    =  08ACD SG%EXC              - 1EDAF SB%TAB(00878) Type=1.2 Nibs=5 Dist=162E2
KEY$   =  1ACA8 SG%KEY              - 1E9AD SB%TAB(00476) Type=1.2 Nibs=5 Dist=03D05
KEY?   =  00712 SB%DVR              - 072E5 JP%SYS(003B6) Type=1.1 Nibs=4 Dist=06BD3
KEYBUF =  2F444 SB%RAM              - 00F1C SB%KEY(00225) Type=0.0 Nibs=5 Offset=     -2
                                    + 02188 SB%DSP(008F8) Type=0.0 Nibs=5
                                    + 15062 MN%ED:(003E2) Type=0.0 Nibs=4 Offset=     28
                                    + 152E8 MN%ED:(00668) Type=0.0 Nibs=5 Offset=     28
                                    + 1535E MN%ED:(006DE) Type=0.0 Nibs=5 Offset=     28
KEYCOD =  1FD22 SB%KCM              - 08E3D SG%EXC(007F5) Type=0.0 Nibs=5
                                    + 14E8C MN%ED:(0020C) Type=0.0 Nibs=5
                                    + 15036 MN%ED:(003B6) Type=0.0 Nibs=5
                                    + 1AC23 SG%KEY(00182) Type=0.0 Nibs=5
KEYDC  =  055E0 SG%LDC              - 08AC3 SG%EXC(0047B) Type=1.2 Nibs=5 Dist=034E3
KEYDEF =  092FA SG%EXC              - 1DF0D JP%TAB(00209) Type=1.2 Nibs=5 Dist=14C13
KEYDEL =  08D2C SG%EXC              -
KEYDWN =  09210 SG%EXC              - 1DF16 JP%TAB(00212) Type=1.2 Nibs=5 Dist=14D06
KEYFND =  08CB8 SG%EXC              - 15193 MN%ED:(00513) Type=0.1 Nibs=5
KEYMRG =  08B8F SG%EXC              -
KEYNAM =  1AC04 SG%KEY              -
```

```
KEYP   =  0375B JP%PR3        - 08AC8 SG%EXC(00480) Type=1.2 Nibs=5 Dist=0536D
KEYPTR =  2F443 SB%RAM        - 00714 SB%DVR(00714) Type=0.0 Nibs=5
                              + 00F23 SB%KEY(0022C) Type=0.0 Nibs=4
                              + 010F0 SB%KEY(003F9) Type=0.0 Nibs=5
                              + 0216B SB%DSP(008DB) Type=0.0 Nibs=5
                              + 092C7 SG%EXC(00C7F) Type=0.0 Nibs=5
                              + 0EF78 MN%UTL(0030A) Type=0.0 Nibs=5
                              + 15050 MN%ED:(003D0) Type=0.0 Nibs=5
                              + 15347 MN%ED:(006C7) Type=0.0 Nibs=5
KEYRD  =  14E11 MN%ED:        - 158FB AB%CLC(003DC) Type=1.1 Nibs=4 Dist=00AEA
KEYSAV =  2F462 SB%RAM        - 000BB SB%DVR(000BB) Type=0.0 Nibs=2
                              + 00DAD SB%KEY(000B6) Type=0.0 Nibs=5
KEYSCN =  00D4D SB%KEY        - 00157 SB%DVR(00157) Type=0.1 Nibs=5
                              + 09218 SG%EXC(00BD0) Type=0.1 Nibs=5
KEYTYP =  15115 MN%ED:        - 09347 SG%EXC(00CFF) Type=0.1 Nibs=5
KILLKY =  15DBD AB%CLC        - 093E4 TI%ERD(00051) Type=0.1 Nibs=5
                              + 16713 AB%ED:(000A1) Type=1.1 Nibs=4 Dist=00956
KMEMCK =  08C4B SG%EXC        - 06DE3 SG%SYS(00A43) Type=1.1 Nibs=4 Dist=01E68
KYARGS =  1AD2F SG%KEY        - 06DAD SG%SYS(00A0D) Type=0.1 Nibs=5
KYDN?  =  00774 SB%DVR        - 1033B MN%CNF(001A9) Type=0.1 Nibs=5
KYFND+ =  08CD4 SG%EXC        - 06DC3 SG%SYS(00A23) Type=1.1 Nibs=4 Dist=01F11
                              + 1AD4C SG%KEY(002AB) Type=0.1 Nibs=5
KYMRG+ =  08BA2 SG%EXC        - 06E1D SG%SYS(00A7D) Type=1.1 Nibs=4 Dist=01D85
KYTYP1 =  1512A MN%ED:        - 1ABB7 SG%KEY(00116) Type=1.1 Nibs=4 Dist=05A8D

LABEL  =  08A24 SG%EXC        - 1EE48 SB%TAB(00911) Type=1.2 Nibs=5 Dist=16424
LABELP =  03E9F JP%PR3        - 02A40 JP%PR1(004A0) Type=1.1 Nibs=4 Dist=0145F
LABLDC =  05702 SG%LDC        -
LAKEYS =  0A36C SG%FXQ        - 08BD7 SG%EXC(0058F) Type=1.1 Nibs=4 Dist=01795
                              + 08CBA SG%EXC(00672) Type=1.1 Nibs=4 Dist=016B2
                              + 1D455 MN%CD:(00BDC) Type=0.1 Nibs=5
LASTFN =  000B4 SB%TAB        - 04FE2 SG%LDC(00084) Type=0.0 Nibs=2
                              + 0F247 AB%EXP(000CF) Type=0.0 Nibs=2
                              + 1AA50 SC%REN(004D4) Type=0.0 Nibs=2
                              + 1AA5D SC%REN(004E1) Type=0.0 Nibs=2
LBLIN# =  2F871 SB%RAM        -
LBLINP =  02A04 JP%PR1        - 02ECA JP%PR2(002A4) Type=1.1 Nibs=3 Dist=004C6
                              + 03C2E JP%PR3(00858) Type=1.1 Nibs=4 Dist=0122A
                              + 03E00 JP%PR3(00A2A) Type=1.1 Nibs=4 Dist=013FC
LBLNAM =  077E7 JP%SYS        - 1A367 SC%SUB(015F5) Type=0.1 Nibs=5
LBLNIF =  02A0D JP%PR1        - 03503 JP%PR3(0012D) Type=1.1 Nibs=4 Dist=00AF6
LBLNPE =  029FD JP%PR1        -
LC     =  07DE6 JP%EXC        -
LCDINI =  00665 SB%DVR        - 01011 SB%KEY(0031A) Type=1.1 Nibs=4 Dist=009AC
LCKMSG =  18D58 MN%LCK        - 18CD3 MN%LCK(0007B) Type=0.0 Nibs=5
LDCEXT =  04F5E SG%LDC        - 178B1 FH%TFM(00E03) Type=0.1 Nibs=5
LDCM10 =  04F6F SG%LDC        - 06AEB SG%SYS(0074B) Type=1.1 Nibs=4 Dist=01B7C
LDCOMP =  04F69 SG%LDC        - 1010A JP%MEM(001AB) Type=0.1 Nibs=5
LDCSET =  05060 SG%LDC        - 02658 JP%PR1(000B8) Type=1.1 Nibs=4 Dist=02A08
                              + 0672E SG%SYS(0038E) Type=1.1 Nibs=4 Dist=016CE
                              + 06A04 SG%SYS(00664) Type=1.1 Nibs=4 Dist=019A4
                              + 072A1 JP%SYS(00372) Type=1.1 Nibs=4 Dist=02241
                              + 07812 JP%SYS(008E3) Type=1.1 Nibs=4 Dist=027B2
                              + 1AAAF SG%KEY(0000E) Type=0.1 Nibs=5
LDCSPC =  2F6C1 SB%RAM        - 04F7F SG%LDC(00021) Type=0.0 Nibs=5
                              + 10117 JP%MEM(001B8) Type=0.0 Nibs=5
                              + 1AAF8 SG%KEY(00057) Type=0.0 Nibs=5
LDSST1 =  04F72 SG%LDC        - 07290 JP%SYS(00361) Type=1.1 Nibs=4 Dist=0231E
LDSST2 =  04F9E SG%LDC        - 072C3 JP%SYS(00394) Type=1.1 Nibs=4 Dist=02325
LEAVE  =  04C01 AB%LEX        -
```

```
LEEWAY =  000D4 TI%EQU        - 012B4 TI%UTL(00183) Type=0.0 Nibs=2
                              + 013DF TI%UTL(002AE) Type=0.0 Nibs=2
                              + 0BFEE AB%FCN(009B5) Type=0.0 Nibs=2
                              + 10C6E MN%CNF(00ADC) Type=0.0 Nibs=2
LEN    =  0BF28 AB%FCN        - 1EB93 SB%TAB(0065C) Type=1.2 Nibs=5 Dist=12C6B
LEN20  =  0BF3F AB%FCN        - 1C874 SG%POK(00386) Type=0.1 Nibs=5
LERRM  =  097AB TI%ERD        - 150F8 MN%ED:(00478) Type=0.1 Nibs=5
LET    =  074BF JP%SYS        - 1EC62 SB%TAB(0072B) Type=1.2 Nibs=5 Dist=177A3
LETDC  =  0526E SG%LDC        - 074B5 JP%SYS(00586) Type=1.2 Nibs=5 Dist=02247
LETP   =  02AD8 JP%PR1        - 074BA JP%SYS(0058B) Type=1.2 Nibs=5 Dist=049E2
LEX:MN =  1F3EA TI%ERM        - 09A22 TI%ERD(0068F) Type=0.0 Nibs=5
LEXBF+ =  10DDF MN%CNF        - 083F7 JP%EXC(0061B) Type=0.1 Nibs=5
                              + 0A1FE SG%FXQ(00688) Type=1.1 Nibs=4 Dist=06BE1
LEXBUF =  10DDB MN%CNF        -
LEXFND =  10F21 MN%CNF        -
LEXPTR =  2F6CF SB%RAM        - 025EB JP%PR1(0004B) Type=0.0 Nibs=5
                              + 03174 JP%PR2(0054E) Type=0.0 Nibs=5
                              + 04D0B AB%LEX(00425) Type=0.0 Nibs=5
LGT12  =  0D1AA JT%MTH        - 0BF6B AB%FCN(00932) Type=1.1 Nibs=4 Dist=0123F
LGT15  =  0D1AE JT%MTH        -
LIF>NB =  17175 FH%TFM        -
LIMITS =  0AC3E AB%REG        -
LIN#A+ =  05125 SG%LDC        - 06A0A SG%SYS(0066A) Type=1.1 Nibs=4 Dist=018E5
LIN#AU =  05122 SG%LDC        - 0FF5A SC%TRC(00513) Type=0.1 Nibs=5
LIN#D+ =  05112 SG%LDC        -
LIN#DC =  05115 SG%LDC        - 072A7 JP%SYS(00378) Type=1.1 Nibs=4 Dist=02192
                              + 0FF0E SC%TRC(004C7) Type=0.1 Nibs=5
LIN#P  =  029E0 JP%PR1        - 17A1F FH%TFM(00F71) Type=0.1 Nibs=5
LIN#P2 =  029E6 JP%PR1        -
LINE#1 =  1A98D SC%REN        -
LINE#? =  02FF7 JP%PR2        - 027E6 JP%PR1(00246) Type=1.0 Nibs=4 Dist=00811
LINEP  =  02620 JP%PR1        - 004AB SB%DVR(004AB) Type=1.0 Nibs=4 Dist=02175
LINEP+ =  02626 JP%PR1        - 0047A SB%DVR(0047A) Type=1.0 Nibs=4 Dist=021AC
LINP   =  02A07 JP%PR1        - 0364E JP%PR3(00278) Type=1.1 Nibs=4 Dist=00C47
                              + 03DEB JP%PR3(00A15) Type=1.1 Nibs=4 Dist=013E4
LINPTP =  031F7 JP%PR2        - 18BE3 SB%IO:(00E27) Type=1.2 Nibs=5 Dist=159EC
LINPUT =  18BE8 SB%IO:        - 1EC59 SB%TAB(00722) Type=1.2 Nibs=5 Dist=06071
LINSKP =  08A05 SG%EXC        - 1B487 MB%IMG(00041) Type=0.1 Nibs=5
LIST   =  06A76 SG%SYS        - 1EC35 SB%TAB(006FE) Type=1.2 Nibs=5 Dist=181BF
LIST*  =  06A7A SG%SYS        -
LISTDC =  05839 SG%LDC        - 06A36 SG%SYS(00696) Type=1.2 Nibs=5 Dist=011FD
                              + 06A6C SG%SYS(006CC) Type=1.2 Nibs=5 Dist=01233
LISTP  =  03B92 JP%PR3        - 06A3B SG%SYS(0069B) Type=1.2 Nibs=5 Dist=02EA9
                              + 06A71 SG%SYS(006D1) Type=1.2 Nibs=5 Dist=02EDF
LN1+12 =  0CD40 JT%MTH        - 0BF7F AB%FCN(00946) Type=1.1 Nibs=4 Dist=00DC1
LN1+15 =  0CD44 JT%MTH        -
LN1+X4 =  0CD5B JT%MTH        -
LN1+XF =  0CD51 JT%MTH        -
LN12   =  0CD7D JT%MTH        - 0BF53 AB%FCN(0091A) Type=1.1 Nibs=4 Dist=00E2A
LN15   =  0CD81 JT%MTH        -
LN30   =  0CD9C JT%MTH        -
LNAP   =  0CF15 JT%MTH        -
LNC10  =  0D195 JT%MTH        -
LNEP26 =  02772 JP%PR1        - 03004 JP%PR2(003DE) Type=1.0 Nibs=4 Dist=00892
LNEP66 =  027EA JP%PR1        - 02AB8 JP%PR1(00518) Type=1.2 Nibs=3 Dist=002CE
                              + 02ABD JP%PR1(0051D) Type=1.2 Nibs=3 Dist=002D3
                              + 02F4D JP%PR2(00327) Type=1.0 Nibs=3 Dist=00763
LNEPLN =  02765 JP%PR1        - 02FFE JP%PR2(003D8) Type=1.0 Nibs=4 Dist=00899
LNNEG  =  0CD90 JT%MTH        -
LNPO6  =  028F9 JP%PR1        - 034A9 JP%PR3(000D3) Type=1.0 Nibs=4 Dist=00BB0
```

```
LNP55  =  02982 JP%PR1        - 02EE2 JP%PR2(002BC) Type=1.0 Nibs=3 Dist=00560
LNPEXT =  02617 JP%PR1        - 17729 FH%TFM(00C7B) Type=0.1 Nibs=5
LNSKP- =  089FF SG%EXC        - 07A31 JP%SYS(00B02) Type=1.1 Nibs=4 Dist=00FCE
                              + 07FFE JP%EXC(00222) Type=1.1 Nibs=4 Dist=00A01
                              + 195EB SC%SUB(00879) Type=0.1 Nibs=5
LOCADR =  0A611 SG%FXQ        - 01348 TI%UTL(00217) Type=0.1 Nibs=5
                              + 067ED SG%SYS(0044D) Type=1.1 Nibs=4 Dist=03E24
LOCFH* =  1719E FH%TFM        -
LOCFHD =  1719A FH%TFM        -
LOCFIH =  17219 FH%TFM        -
LOCFIL =  1721D FH%TFM        -
LOCK   =  18C62 MN%LCK        - 1DF31 JP%TAB(0022D) Type=1.2 Nibs=5 Dist=052CF
LOCKD? =  18C98 MN%LCK        - 0064E SB%DVR(0064E) Type=0.1 Nibs=5
LOCKWD =  2F7B2 SB%RAM        - 10738 MN%CNF(005A6) Type=0.0 Nibs=5
                              + 18C8B MN%LCK(00033) Type=0.0 Nibs=5
                              + 18CAB MN%LCK(00053) Type=0.0 Nibs=5
                              + 18D3A MN%LCK(000E2) Type=0.0 Nibs=5
LOG    =  0BF4A AB%FCN        - 1EAB2 SB%TAB(0057B) Type=1.2 Nibs=5 Dist=12B68
                              + 1EABB SB%TAB(00584) Type=1.2 Nibs=5 Dist=12B71
LOG10  =  0BF62 AB%FCN        - 1DF28 JP%TAB(00224) Type=1.2 Nibs=5 Dist=11FC6
                              + 1EACD SB%TAB(00596) Type=1.2 Nibs=5 Dist=12B6B
LOGIC  =  0B869 AB%FCN        -
LOGP1  =  0BF76 AB%FCN        - 1DF3A JP%TAB(00236) Type=1.2 Nibs=5 Dist=11FC4
LOOPST =  2F7AC SB%RAM        -
LOWERC =  15243 MN%ED:        -
LPRNCK =  033C4 JP%PR2        - 035F9 JP%PR3(00223) Type=1.1 Nibs=3 Dist=00235
LPRP   =  0F23C AB%EXP        - 1EB9C SB%TAB(00665) Type=1.2 Nibs=5 Dist=0F960
LR     =  0E3FC PM%STA        - 1EC08 SB%TAB(006D1) Type=1.2 Nibs=5 Dist=1080C
LRDC   =  05493 SG%LDC        - 0E3F2 PM%STA(005C3) Type=1.2 Nibs=5 Dist=08F5F
LRP    =  039C6 JP%PR3        - 0E3F7 PM%STA(005C8) Type=1.2 Nibs=5 Dist=0AA31
LSLEEP =  006CD SB%DVR        - 00191 SB%DVR(00191) Type=1.0 Nibs=3 Dist=0053C
                              + 0ED91 MN%UTL(00123) Type=0.1 Nibs=5
LSTADR =  0FFC6 JP%MEM        - 01285 TI%UTL(00154) Type=0.1 Nibs=5
                              + 08594 JP%EXC(007B8) Type=1.1 Nibs=4 Dist=07A32
                              + 0BFB8 AB%FCN(0097F) Type=1.1 Nibs=4 Dist=0400E
                              + 14ACB SC%DAT(01216) Type=1.1 Nibs=4 Dist=04B05
LSTKEY =  1ACF4 SG%KEY        - 06A67 SG%SYS(006C7) Type=0.1 Nibs=5
LSTLEN =  06C27 SG%SYS        - 05375 SG%LDC(00417) Type=1.0 Nibs=4 Dist=018B2
LSTLN+ =  06C1E SG%SYS        - 1ABBD SG%KEY(0011C) Type=0.1 Nibs=5
LXBFID =  00BFC TI%EQU        -
LXFND  =  0979D TI%ERD        - 0813D JP%EXC(00361) Type=1.1 Nibs=4 Dist=01660
                              + 0A8E5 SG%FXQ(00D6F) Type=1.1 Nibs=4 Dist=01148
                              + 12474 JP%POL(00159) Type=0.1 Nibs=5
LXSPDT =  1E543 SB%TAB        -
LXTXTT =  1EE9F SB%TAB        - 150B8 MN%ED:(00438) Type=0.0 Nibs=5
LXTYPT =  1FE74 AB%LXT        - 0499C AB%LEX(000B6) Type=0.0 Nibs=5 Offset=    -99

MAIN05 =  00338 SB%DVR        -
MAIN10 =  0037E SB%DVR        - 0170B SB%CMD(000E4) Type=1.0 Nibs=4 Dist=0138D
                              + 02F63 JP%PR2(0033D) Type=1.0 Nibs=4 Dist=02BE5
MAIN30 =  0037E SB%DVR        - 10151 JP%MEM(001F2) Type=0.1 Nibs=5
MAINEN =  2F571 SB%RAM        - 084E6 JP%EXC(0070A) Type=0.0 Nibs=2
                              + 0A616 SG%FXQ(00AA0) Type=0.0 Nibs=5
                              + 0A82F SG%FXQ(00CB9) Type=0.0 Nibs=2
                              + 118C3 SC%FIL(00870) Type=0.0 Nibs=5
                              + 1D947 MN%CD:(010CE) Type=0.0 Nibs=5
MAINLP =  002FD SB%DVR        - 010D8 SB%KEY(003E1) Type=1.0 Nibs=4 Dist=00DDB
                              + 02896 JP%PR1(002F6) Type=1.0 Nibs=4 Dist=02599
                              + 07256 JP%SYS(00327) Type=1.0 Nibs=4 Dist=06F59
MAINST =  2F558 SB%RAM        - 0A012 SG%FXQ(0049C) Type=0.0 Nibs=5
```

```
                                + 10753 MN%CNF(005C1) Type=0.0 Nibs=5
                                + 10CB6 MN%CNF(00B24) Type=0.0 Nibs=4
                                + 10CC6 MN%CNF(00B34) Type=0.0 Nibs=5
                                + 1A239 SC%SUB(014C7) Type=0.0 Nibs=5
MAINT  =  1E59F SB%TAB          - 0508D SG%LDC(0012F) Type=0.0 Nibs=5
                                + 08197 JP%EXC(003BB) Type=0.0 Nibs=5
                                + 0862C JP%EXC(00850) Type=0.0 Nibs=5 Offset=      3
                                + 0F255 AB%EXP(000DD) Type=0.0 Nibs=5 Offset=      3
MAINTS =  1E537 SB%TAB          - 1DD0A JP%TAB(00006) Type=1.2 Nibs=5 Dist=0082D
MAKE1  =  0DACE SM%MTH          - 0D1BB JT%MTH(00E94) Type=1.1 Nibs=4 Dist=00913
                                + 0D2BD JT%MTH(00F96) Type=1.1 Nibs=4 Dist=00811
MAKEBF =  01751 SB%CMD          - 0015E SB%DVR(0015E) Type=0.1 Nibs=5
                                + 02622 JP%PR1(00082) Type=1.1 Nibs=4 Dist=00ED1
                                + 1875B SB%IO:(0099F) Type=0.1 Nibs=5
MANSTK =  196D3 SC%SUB          - 07D9C JP%SYS(00E6D) Type=0.1 Nibs=5
MAX    =  0E8AA PM%STA          - 1EBB7 SB%TAB(00680) Type=1.2 Nibs=5 Dist=1030D
MAXCMD =  2F976 SB%RAM          - 00233 SB%DVR(00233) Type=0.0 Nibs=2
                                + 016D3 SB%CMD(000AC) Type=0.0 Nibs=5
                                + 0180E SB%CMD(001E7) Type=0.0 Nibs=5
                                + 15521 AB%CLC(00002) Type=0.0 Nibs=5
                                + 155EF AB%CLC(000D0) Type=0.0 Nibs=5
MAXRL  =  0B76E AB%FCN          - 1E96E SB%TAB(00437) Type=1.2 Nibs=5 Dist=13200
MBOX^  =  2F7A9 SB%RAM          -
MEAN   =  0E2D1 PM%STA          - 1DF55 JP%TAB(00251) Type=1.2 Nibs=5 Dist=0FC84
                                + 1EB27 SB%TAB(005F0) Type=1.2 Nibs=5 Dist=10856
MEM    =  0BF92 AB%FCN          - 1DF5E JP%TAB(0025A) Type=1.2 Nibs=5 Dist=11FCC
MEMBER =  1B098 AB%UTL          - 04007 SB%EXP(00032) Type=0.1 Nibs=5
                                + 08E33 SG%EXC(007EB) Type=0.1 Nibs=5
                                + 157B8 AB%CLC(00299) Type=1.1 Nibs=4 Dist=058E0
                                + 15A49 AB%CLC(0052A) Type=1.1 Nibs=4 Dist=0564F
                                + 1607D AB%CLC(00B5E) Type=1.1 Nibs=4 Dist=0501B
                                + 16122 AB%CLC(00C03) Type=1.1 Nibs=4 Dist=04F76
                                + 16185 AB%CLC(00C66) Type=1.1 Nibs=4 Dist=04F13
                                + 162A2 AB%CLC(00D83) Type=1.1 Nibs=4 Dist=04DF6
                                + 164C2 AB%BLD(00196) Type=1.1 Nibs=4 Dist=04BD6
                                + 167E3 AB%ED:(00171) Type=1.1 Nibs=4 Dist=048B5
MEMCKL =  012A5 TI%UTL          - 084DA JP%EXC(006FE) Type=1.1 Nibs=4 Dist=07235
                                + 08693 SG%EXC(0004B) Type=1.1 Nibs=4 Dist=073EE
                                + 0A1D9 SG%FXQ(00663) Type=0.1 Nibs=5
                                + 1234C JP%POL(00031) Type=0.1 Nibs=5
                                + 170AE FH%TFM(00600) Type=0.1 Nibs=5
                                + 17A00 FH%TFM(00F52) Type=0.1 Nibs=5
                                + 17A81 MN%FRP(0004E) Type=0.1 Nibs=5
                                + 1D22B MN%CD:(009B2) Type=0.1 Nibs=5
MEMCL+ =  012A7 TI%UTL          - 08C4D SG%EXC(00605) Type=1.1 Nibs=4 Dist=079A6
                                + 08C89 SG%EXC(00641) Type=1.1 Nibs=4 Dist=079E2
                                + 119A1 SC%FIL(0094E) Type=0.1 Nibs=5
                                + 17695 FH%TFM(00BE7) Type=0.1 Nibs=5
MEMER* =  0945B TI%ERD          -
MEMERR =  0944D TI%ERD          - 02D4A JP%PR2(00124) Type=1.0 Nibs=4 Dist=06703
                                + 04606 SB%EXP(00631) Type=1.0 Nibs=4 Dist=04E47
                                + 05441 SG%LDC(004E3) Type=1.0 Nibs=4 Dist=0400C
                                + 05FB3 SB%EXD(00691) Type=1.0 Nibs=4 Dist=0349A
                                + 0AD61 AB%REG(00169) Type=1.0 Nibs=4 Dist=01914
                                + 0C1CB AB%FCN(00B92) Type=1.0 Nibs=4 Dist=02D7E
                                + 0F5DC AB%ASN(00002) Type=1.0 Nibs=4 Dist=0618F
MEMERX =  0944F TI%ERD          - 00482 SB%DVR(00482) Type=0.1 Nibs=5
MEMERj =  17F80 SB%IO:          - 17BBB MN%FRP(00188) Type=1.0 Nibs=3 Dist=003C5
                                + 19A77 SC%SUB(00D05) Type=1.0 Nibs=4 Dist=01AF7
                                + 1A537 SB%EXC(00163) Type=1.0 Nibs=4 Dist=025B7
```

```
                                + 1BAA0 MB%IMG(0065A) Type=1.0 Nibs=4 Dist=03B20
                                + 1C43B SB%GPH(00076) Type=1.0 Nibs=4 Dist=044BB
MEMERk =  0FF33 SC%TRC          -
MERGDC =  05844 SG%LDC          - 06C9C SG%SYS(008FC) Type=1.2 Nibs=5 Dist=01458
MERGE  =  06CA6 SG%SYS          - 1DF67 JP%TAB(00263) Type=1.2 Nibs=5 Dist=172C1
MERGEP =  03B85 JP%PR3          - 06CA1 SG%SYS(00901) Type=1.2 Nibs=5 Dist=0311C
MESSG  =  0CC17 JT%MTH          -
MFDEVC =  012E5 TI%UTL          - 082C5 JP%EXC(004E9) Type=1.1 Nibs=4 Dist=06FE0
MFDVC+ =  012DF TI%UTL          - 08306 JP%EXC(0052A) Type=1.1 Nibs=4 Dist=07027
                                + 0836B JP%EXC(0058F) Type=1.1 Nibs=4 Dist=0708C
                                + 08419 JP%EXC(0063D) Type=1.1 Nibs=4 Dist=0713A
MFDVC- =  012E8 TI%UTL          -
MFER42 =  0962C TI%ERD          - 18B56 SB%IO:(00D9A) Type=0.1 Nibs=5
MFERR  =  09393 TI%ERD          - 001DF SB%DVR(001DF) Type=0.0 Nibs=5
                                + 05822 SG%LDC(008C4) Type=1.0 Nibs=4 Dist=03B71
                                + 06992 SG%SYS(005F2) Type=1.0 Nibs=4 Dist=02A01
                                + 07768 JP%SYS(00839) Type=1.0 Nibs=4 Dist=01C2B
                                + 080C6 JP%EXC(002EA) Type=1.0 Nibs=4 Dist=012CD
                                + 08C55 SG%EXC(0060D) Type=1.0 Nibs=3 Dist=0073E
                                + 08D74 SG%EXC(0072C) Type=1.0 Nibs=3 Dist=0061F
                                + 09007 SG%EXC(009BF) Type=1.0 Nibs=3 Dist=0038C
                                + 09088 SG%EXC(00A40) Type=1.0 Nibs=3 Dist=0030B
                                + 0A320 SG%FXQ(007AA) Type=1.0 Nibs=4 Dist=00F8D
                                + 0B02A AB%REG(00432) Type=1.0 Nibs=4 Dist=01C97
                                + 0BF21 AB%FCN(008E8) Type=1.0 Nibs=4 Dist=02B8E
                                + 0E12C PM%STA(002FD) Type=1.0 Nibs=4 Dist=04D99
                                + 0FFC2 JP%MEM(00063) Type=1.0 Nibs=4 Dist=06C2F
MFERR* =  093F1 TI%ERD          - 16B51 FH%TFM(000A3) Type=0.1 Nibs=5
                                + 18961 SB%IO:(00BA5) Type=0.1 Nibs=5
MFERR- =  093EE TI%ERD          - 02F5D JP%PR2(00337) Type=1.1 Nibs=4 Dist=06491
MFERRO =  0E128 PM%STA          - 1B4F0 MB%IMG(000AA) Type=0.1 Nibs=5
MFERRS =  0939E TI%ERD          - 16B72 FH%TFM(000C4) Type=0.1 Nibs=5
MFERRj =  17CCC SB%RD:          - 199A3 SC%SUB(00C31) Type=1.0 Nibs=4 Dist=01CD7
                                + 1A771 SC%REN(001F5) Type=1.0 Nibs=4 Dist=02AA5
                                + 1C77C SG%POK(0028E) Type=1.0 Nibs=4 Dist=04AB0
                                + 1D048 MN%CD:(007CF) Type=1.0 Nibs=4 Dist=0537C
MFERsp =  0940D TI%ERD          -
MFLG=0 =  13DA1 SC%DAT          -
MFLG=X =  13DA3 SC%DAT          - 0960F TI%ERD(0027C) Type=0.1 Nibs=5
MFWRN  =  093BC TI%ERD          - 10EA8 MN%CNF(00D16) Type=1.1 Nibs=4 Dist=07AEC
                                + 15953 AB%CLC(00434) Type=0.1 Nibs=5
                                + 1D002 MN%CD:(00789) Type=0.1 Nibs=5
                                + 1D016 MN%CD:(0079D) Type=0.1 Nibs=5
MFWRNQ =  093C5 TI%ERD          - 0CC9F JT%MTH(00978) Type=1.1 Nibs=4 Dist=038DA
                                + 161F1 AB%CLC(00CD2) Type=0.1 Nibs=5
                                + 16218 AB%CLC(00CF9) Type=0.1 Nibs=5
                                + 1625B AB%CLC(00D3C) Type=0.1 Nibs=5
                                + 1626F AB%CLC(00D50) Type=0.1 Nibs=5
                                + 172E7 FH%TFM(00839) Type=0.1 Nibs=5
MFWRQ8 =  093C3 TI%ERD          - 04749 SB%EXP(00774) Type=1.1 Nibs=4 Dist=04C7A
                                + 06C48 SG%SYS(008A8) Type=1.1 Nibs=4 Dist=0277B
                                + 0A091 SG%FXQ(0051B) Type=1.1 Nibs=4 Dist=00CCE
                                + 17FDD SB%IO:(00221) Type=0.1 Nibs=5
                                + 1C2D9 MB%USG(00835) Type=0.1 Nibs=5
MFerrj =  1A76F SC%REN          -
MGOSUB =  1AF01 MN%GSB          -
MIN    =  0E8C5 PM%STA          - 1EBAE SB%TAB(00677) Type=1.2 Nibs=5 Dist=102E9
MINRL  =  0B777 AB%FCN          - 1DF70 JP%TAB(0026C) Type=1.2 Nibs=5 Dist=127F9
MINUS  =  0B650 AB%FCN          - 1EA34 SB%TAB(004FD) Type=1.2 Nibs=5 Dist=133E4
MLFFLG =  2F870 SB%RAM          - 06A7C SG%SYS(006DC) Type=0.0 Nibs=5
```

```
                                + 13DA5 SC%DAT(004F0) Type=0.0 Nibs=5
                                + 13DFE SC%DAT(00549) Type=0.0 Nibs=5
                                + 1853A SB%IO:(0077E) Type=0.0 Nibs=5
                                + 18549 SB%IO:(0078D) Type=0.0 Nibs=5
                                + 19EAE SC%SUB(0113C) Type=0.0 Nibs=5
MMCORJ =  1AF57 MN%GSB          -
MOD    =  0C117 AB%FCN          - 1E9B6 SB%TAB(0047F) Type=1.2 Nibs=5 Dist=1289F
MOD12  =  0C792 JT%MTH          - 0C11F AB%FCN(00AE6) Type=1.1 Nibs=3 Dist=00673
MOD15  =  0C796 JT%MTH          -
MOVCUR =  0233E SB%DSP          -
MOVE*M =  01308 TI%UTL          - 013A6 TI%UTL(00275) Type=1.1 Nibs=3 Dist=0009E
                                + 176B9 FH%TFM(00C0B) Type=0.1 Nibs=5
MOVED0 =  1B0F4 AB%UTL          - 05BF8 SB%EXD(002D6) Type=0.1 Nibs=5
                                + 060E7 SB%EXD(007C5) Type=0.1 Nibs=5
                                + 1B4D8 MB%IMG(00092) Type=1.1 Nibs=3 Dist=003E4
MOVED1 =  1B101 AB%UTL          - 15856 AB%CLC(00337) Type=1.1 Nibs=4 Dist=058AB
MOVED2 =  1B104 AB%UTL          - 08C30 SG%EXC(005E8) Type=0.1 Nibs=5
                                + 10CBF MN%CNF(00B2D) Type=0.1 Nibs=5
                                + 119C1 SC%FIL(0096E) Type=0.1 Nibs=5
                                + 15F90 AB%CLC(00A71) Type=1.1 Nibs=4 Dist=05174
                                + 1ADF7 SB%VAL(00089) Type=1.1 Nibs=3 Dist=0030D
MOVED3 =  1B109 AB%UTL          - 0160F TI%UTL(004DE) Type=0.1 Nibs=5
                                + 0B4F4 AB%REG(008FC) Type=0.1 Nibs=5
                                + 14B5B SC%DAT(012A6) Type=1.1 Nibs=4 Dist=065AE
                                + 17AB7 MN%FRP(00084) Type=1.1 Nibs=4 Dist=03652
                                + 18BDA SB%IO:(00E1E) Type=1.0 Nibs=4 Dist=0252F
                                + 1995E SC%SUB(00BEC) Type=1.0 Nibs=4 Dist=017AB
                                + 1D6CD MN%CD:(00E54) Type=1.1 Nibs=4 Dist=025C4
MOVEDA =  1B0FA AB%UTL          -
MOVEDD =  1B106 AB%UTL          - 0C2AC AB%FCN(00C73) Type=0.1 Nibs=5
MOVEDM =  1B0EE AB%UTL          - 01315 TI%UTL(001E4) Type=0.1 Nibs=5
MOVEU0 =  1B162 AB%UTL          - 083C5 JP%EXC(005E9) Type=0.1 Nibs=5
MOVEU1 =  1B16F AB%UTL          - 08CA0 SG%EXC(00658) Type=0.1 Nibs=5
                                + 0A1E7 SG%FXQ(00671) Type=0.1 Nibs=5
                                + 15DA6 AB%CLC(00887) Type=1.1 Nibs=4 Dist=053C9
                                + 15F60 AB%CLC(00A41) Type=1.1 Nibs=4 Dist=0520F
                                + 1690F AB%ED:(0029D) Type=1.1 Nibs=4 Dist=04860
MOVEU2 =  1B172 AB%UTL          - 01856 SB%CMD(0022F) Type=0.1 Nibs=5
                                + 06D5C SG%SYS(009BC) Type=0.1 Nibs=5
                                + 0B50F AB%REG(00917) Type=0.1 Nibs=5
MOVEU3 =  1B177 AB%UTL          - 00460 SB%DVR(00460) Type=0.1 Nibs=5
                                + 09340 SG%EXC(00CF8) Type=0.1 Nibs=5
                                + 156A7 AB%CLC(00188) Type=1.0 Nibs=4 Dist=05AD0
                                + 16953 AB%ED:(002E1) Type=1.1 Nibs=4 Dist=04824
                                + 16A51 AB%ED:(003DF) Type=1.1 Nibs=4 Dist=04726
                                + 17AE3 MN%FRP(000B0) Type=1.1 Nibs=4 Dist=03694
                                + 18B91 SB%IO:(00DD5) Type=1.1 Nibs=4 Dist=025E6
                                + 19BEE SC%SUB(00E7C) Type=1.1 Nibs=4 Dist=01589
MOVEU4 =  1B174 AB%UTL          - 06E32 SG%SYS(00A92) Type=0.1 Nibs=5
MOVEUA =  1B168 AB%UTL          - 028C9 JP%PR1(00329) Type=0.1 Nibs=5
                                + 08D53 SG%EXC(0070B) Type=0.1 Nibs=5
                                + 10C3E MN%CNF(00AAC) Type=0.1 Nibs=5
                                + 11A86 SC%FIL(00A33) Type=0.1 Nibs=5
MOVEUM =  1B15C AB%UTL          - 01324 TI%UTL(001F3) Type=0.1 Nibs=5
MOVNXT =  08B87 SG%EXC          - 1A530 SB%EXC(0015C) Type=0.1 Nibs=5
MOved3 =  0B4F2 AB%REG          - 05CF0 SB%EXD(003CE) Type=1.1 Nibs=4 Dist=05802
                                + 084F8 JP%EXC(0071C) Type=1.1 Nibs=4 Dist=02FFA
                                + 08F76 SG%EXC(0092E) Type=1.1 Nibs=4 Dist=0257C
                                + 0B6EF AB%FCN(000B6) Type=1.1 Nibs=3 Dist=001FD
                                + 0CCC9 JT%MTH(009A2) Type=1.1 Nibs=4 Dist=017D7
```

```
                                    + 0F446 AB%EXP(002CE) Type=1.1 Nibs=4 Dist=03F54
                                    + 0F73B AB%ASN(00161) Type=1.1 Nibs=4 Dist=04249
                                    + 0F768 AB%ASN(0018E) Type=1.1 Nibs=4 Dist=04276
                                    + 10996 MN%CNF(00804) Type=1.1 Nibs=4 Dist=054A4
                                    + 11C19 SC%FIL(00BC6) Type=1.1 Nibs=4 Dist=06727
MOveu3 =  0933E SG%EXC              - 01566 TI%UTL(00435) Type=1.1 Nibs=4 Dist=07DD8
                                    + 02063 SB%DSP(007D3) Type=1.1 Nibs=4 Dist=072DB
                                    + 0998E TI%ERD(005FB) Type=1.1 Nibs=3 Dist=00650
                                    + 0CC7F JT%MTH(00958) Type=1.1 Nibs=4 Dist=03941
MP1-12 =  0C436 JT%MTH              -
MP15S  =  0C440 JT%MTH              - 0E1B7 PM%STA(00388) Type=1.1 Nibs=4 Dist=01D77
MP2-12 =  0C432 JT%MTH              - 0B67B AB%FCN(00042) Type=1.1 Nibs=4 Dist=00DB7
                                    + 0EEF3 MN%UTL(00285) Type=1.1 Nibs=4 Dist=02AC1
                                    + 12E8C MN%TM:(00942) Type=1.1 Nibs=4 Dist=06A5A
                                    + 13213 MN%TM:(00CC9) Type=1.1 Nibs=4 Dist=06DE1
MP2-15 =  0C43A JT%MTH              - 0B6AE AB%FCN(00075) Type=1.1 Nibs=4 Dist=00D8C
                                    + 0D22C JT%MTH(00F05) Type=1.1 Nibs=4 Dist=00DF2
                                    + 0D9DA SM%MTH(005A5) Type=1.1 Nibs=4 Dist=015A0
MPOP1N =  0BD8D AB%FCN              -
MPOP2N =  0BD54 AB%FCN              -
MPY    =  0ECBB MN%UTL              - 0EBB5 MN%BP:(001C4) Type=1.1 Nibs=3 Dist=00106
                                    + 12CE2 MN%TM:(00798) Type=1.0 Nibs=4 Dist=04027
MSIZ++ =  1040D MN%CNF              -
MSIZE  =  10407 MN%CNF              -
MSIZE+ =  1040A MN%CNF              - 0A4A5 SG%FXQ(0092F) Type=1.1 Nibs=4 Dist=05F65
                                    + 0FFCA JP%MEM(0006B) Type=1.1 Nibs=3 Dist=00440
                                    + 17BAE MN%FRP(0017B) Type=1.1 Nibs=4 Dist=077A4
MSN12  =  0D553 SM%MTH              - 0E372 PM%STA(00543) Type=1.1 Nibs=4 Dist=00E1F
MSN15  =  0D557 SM%MTH              - 0B8E4 AB%FCN(002AB) Type=1.1 Nibs=4 Dist=01C73
                                    + 0C8BF JT%MTH(00598) Type=1.1 Nibs=4 Dist=00C98
                                    + 0D281 JT%MTH(00F5A) Type=1.1 Nibs=3 Dist=002D6
MSPARe =  02E5C JP%PR2              -
MSTKD1 =  1953C SC%SUB              -
MTADDR =  08195 JP%EXC              - 1506F MN%ED:(003EF) Type=0.1 Nibs=5
MTADR+ =  081A1 JP%EXC              - 050CE SG%LDC(00170) Type=1.1 Nibs=4 Dist=030D3
MTHSTK =  2F599 SB%RAM              - 0B17E AB%REG(00586) Type=0.0 Nibs=5
                                    + 0B1E1 AB%REG(005E9) Type=0.0 Nibs=5
                                    + 0B285 AB%REG(0068D) Type=0.0 Nibs=5
                                    + 0B4D9 AB%REG(008E1) Type=0.0 Nibs=5
                                    + 0B4FD AB%REG(00905) Type=0.0 Nibs=5
                                    + 0B529 AB%REG(00931) Type=0.0 Nibs=5
                                    + 0E0CE PM%STA(0029F) Type=0.0 Nibs=4
                                    + 0F188 AB%EXP(00010) Type=0.0 Nibs=5
                                    + 0F7A3 AB%ASN(001C9) Type=0.0 Nibs=5
                                    + 0F933 AB%ASN(00359) Type=0.0 Nibs=5
                                    + 1079A MN%CNF(00608) Type=0.0 Nibs=5
                                    + 13DDD SC%DAT(00528) Type=0.0 Nibs=5
                                    + 14890 SC%DAT(00FDB) Type=0.0 Nibs=5
                                    + 14C30 SC%DAT(0137B) Type=0.0 Nibs=5
                                    + 15E07 AB%CLC(008E8) Type=0.0 Nibs=5
                                    + 15E5C AB%CLC(0093D) Type=0.0 Nibs=5
                                    + 15F77 AB%CLC(00A58) Type=0.0 Nibs=5
                                    + 162DB AB%CLC(00DBC) Type=0.0 Nibs=5
                                    + 16400 AB%BLD(000D4) Type=0.0 Nibs=5
                                    + 1655D AB%BLD(00231) Type=0.0 Nibs=5
                                    + 19006 SC%SUB(00294) Type=0.0 Nibs=5
                                    + 1915A SC%SUB(003E8) Type=0.0 Nibs=5
                                    + 19541 SC%SUB(007CF) Type=0.0 Nibs=5
                                    + 19550 SC%SUB(007DE) Type=0.0 Nibs=5
                                    + 19D78 SC%SUB(01006) Type=0.0 Nibs=5
```

```
                                + 19F42 SC%SUB(011D0) Type=0.0 Nibs=5
                                + 1ADFF SB%VAL(00091) Type=0.0 Nibs=5
                                + 1AE50 SB%VAL(000E2) Type=0.0 Nibs=4
MULTF  =  0C446 JT%MTH          - 0D8D1 SM%MTH(0049C) Type=1.1 Nibs=4 Dist=0148B
                                + 0DCB3 SM%MTH(0087E) Type=1.1 Nibs=4 Dist=0186D
MULTPY =  0B672 AB%FCN          - 1EA3D SB%TAB(00506) Type=1.2 Nibs=5 Dist=133CB
MVMEM  =  01363 TI%UTL          -
MVMEM+ =  0133C TI%UTL          - 06976 SG%SYS(005D6) Type=1.1 Nibs=4 Dist=0563A
                                + 173FA FH%TFM(0094C) Type=0.1 Nibs=5
                                + 1D95B MN%CD:(010E2) Type=0.1 Nibs=5
MfErr  =  0FFC0 JP%MEM          - 0F854 AB%ASN(0027A) Type=1.0 Nibs=3 Dist=0076C
                                + 0F950 AB%ASN(00376) Type=1.0 Nibs=3 Dist=00670
                                + 115BB SC%FIL(00568) Type=1.0 Nibs=4 Dist=015FB
                                + 12DE5 MN%TM:(0089B) Type=1.0 Nibs=4 Dist=02E25
                                + 13FF6 SC%DAT(00741) Type=1.0 Nibs=4 Dist=04036
                                + 16212 AB%CLC(00CF3) Type=1.0 Nibs=4 Dist=06252
                                + 17CCE SB%RD:(00107) Type=1.0 Nibs=4 Dist=07D0E
Moved3 =  0160D TI%UTL          - 01EBD SB%DSP(0062D) Type=1.1 Nibs=4 Dist=008B0
Moveu3 =  156A5 AB%CLC          - 0F790 AB%ASN(001B6) Type=1.1 Nibs=4 Dist=05F15
                                + 0F79D AB%ASN(001C3) Type=1.1 Nibs=4 Dist=05F08
                                + 10A77 MN%CNF(008E5) Type=1.1 Nibs=4 Dist=04C2E
                                + 10BCC MN%CNF(00A3A) Type=1.1 Nibs=4 Dist=04AD9
                                + 12384 JP%POL(00069) Type=1.1 Nibs=4 Dist=03321

NAME   =  0A332 SG%FXQ          - 1EC47 SB%TAB(00710) Type=1.2 Nibs=5 Dist=14915
NAMEDC =  054E5 SG%LDC          - 0A328 SG%FXQ(007B2) Type=1.2 Nibs=5 Dist=04E43
                                + 0A436 SG%FXQ(008C0) Type=1.2 Nibs=5 Dist=04F51
NAMEP  =  03B6E JP%PR3          - 0A32D SG%FXQ(007B7) Type=1.2 Nibs=5 Dist=067BF
NAN    =  0B7B3 AB%FCN          - 1DF79 JP%TAB(00275) Type=1.2 Nibs=5 Dist=127C6
NASSAU =  17B33 MN%FRP          - 1DE11 JP%TAB(0010D) Type=1.2 Nibs=5 Dist=062DE
NEEDSC =  2F94A SB%RAM          - 01E71 SB%DSP(005E1) Type=0.0 Nibs=4
                                + 02268 SB%DSP(009D8) Type=0.0 Nibs=5
                                + 14C8C MN%ED:(0000C) Type=0.0 Nibs=5
                                + 1C4B1 SB%GPH(000EC) Type=0.0 Nibs=4
NEWSTM =  15586 AB%CLC          - 1698A AB%ED:(00318) Type=1.0 Nibs=4 Dist=01404
NEWVAR =  0F829 AB%ASN          - 13E71 SC%DAT(005BC) Type=1.1 Nibs=4 Dist=04648
                                + 18F51 SC%SUB(001DF) Type=0.1 Nibs=5
NEXT   =  0884C SG%EXC          - 1EC86 SB%TAB(0074F) Type=1.2 Nibs=5 Dist=1643A
NEXTst =  0EDC6 MN%UTL          - 0EA90 MN%BP:(0009F) Type=1.0 Nibs=3 Dist=00336
                                + 11660 SC%FIL(0060D) Type=1.0 Nibs=4 Dist=0289A
                                + 12DC8 MN%TM:(0087E) Type=1.0 Nibs=4 Dist=04002
                                + 14386 SC%DAT(00AD1) Type=1.0 Nibs=4 Dist=055C0
NOKEY+ =  0246D SB%DSP          -
NOKEYS =  0EF76 MN%UTL          - 0247F SB%DSP(00BEF) Type=0.1 Nibs=5
                                + 1D083 MN%CD:(0080A) Type=0.1 Nibs=5
NOMEM  =  0F5DA AB%ASN          - 0E26C PM%STA(0043D) Type=1.0 Nibs=4 Dist=0136E
                                + 0F0D5 MN%UTL(00467) Type=1.0 Nibs=3 Dist=00505
                                + 0F1DA AB%EXP(00062) Type=1.0 Nibs=3 Dist=00400
                                + 0FF35 SC%TRC(004EE) Type=1.0 Nibs=4 Dist=0095B
                                + 11BF5 SC%FIL(00BA2) Type=1.0 Nibs=4 Dist=0261B
                                + 143CE SC%DAT(00B19) Type=1.0 Nibs=4 Dist=04DF4
NOPRGM =  076DB JP%SYS          - 0034A SB%DVR(0034A) Type=1.1 Nibs=4 Dist=07391
                                + 09115 SG%EXC(00ACD) Type=1.1 Nibs=4 Dist=01A3A
                                + 1631B AB%CLC(00DFC) Type=0.1 Nibs=5
NORDIM =  0AE2D AB%REG          -
NOSCRL =  14C8A MN%ED:          - 004FD SB%DVR(004FD) Type=0.1 Nibs=5
                                + 010CA SB%KEY(003D3) Type=0.1 Nibs=5
                                + 06580 SG%SYS(001E0) Type=0.1 Nibs=5
                                + 072CF JP%SYS(003A0) Type=0.1 Nibs=5
                                + 10E8D MN%CNF(00CFB) Type=1.1 Nibs=4 Dist=03DFD
```

```
                                + 18B02 SB%IO:(00D46) Type=1.1 Nibs=4 Dist=03E78
                                + 1CED7 MN%CD:(0065E) Type=0.1 Nibs=5
NOSTAT =  0B024 AB%REG          -
NOT    =  0B660 AB%FCN          - 1EA2B SB%TAB(004F4) Type=1.2 Nibs=5 Dist=133CB
NOTIME =  01AAE SB%DSP          -
NOTMCH =  19476 SC%SUB          -
NOXFN  =  0C31D AB%FCN          -
NRMCON =  161AF AB%CLC          -
NRMLAB =  0C6D5 JT%MTH          -
NRMLCD =  0C956 JT%MTH          -
NTOKEN =  0493B AB%LEX          - 02BC9 JP%PR1(00629) Type=1.0 Nibs=4 Dist=01D72
                                + 03885 JP%PR3(004AF) Type=1.0 Nibs=4 Dist=010B6
                                + 041B2 SB%EXP(001DD) Type=1.1 Nibs=3 Dist=00789
                                + 0460F SB%EXP(0063A) Type=1.0 Nibs=3 Dist=0032C
                                + 0487E SB%EXP(008A9) Type=1.1 Nibs=3 Dist=000BD
                                + 04893 SB%EXP(008BE) Type=1.1 Nibs=3 Dist=000A8
                                + 159C1 AB%CLC(004A2) Type=0.1 Nibs=5
                                + 15BC3 AB%CLC(006A4) Type=0.1 Nibs=5
NTOKNL =  048E6 AB%LEX          - 025C1 JP%PR1(00021) Type=1.1 Nibs=4 Dist=02325
                                + 029E2 JP%PR1(00442) Type=1.1 Nibs=4 Dist=01F04
                                + 02A09 JP%PR1(00469) Type=1.1 Nibs=4 Dist=01EDD
                                + 03BD9 JP%PR3(00803) Type=1.1 Nibs=4 Dist=00D0D
NULLBF =  0188B SB%CMD          - 017FF SB%CMD(001D8) Type=0.0 Nibs=5
NULLP  =  07999 JP%SYS          - 0A503 SG%FXQ(0098D) Type=1.1 Nibs=4 Dist=02B6A
                                + 10054 JP%MEM(000F5) Type=0.1 Nibs=5
NUM    =  0A9BE SB%FCN          - 1EB5D SB%TAB(00626) Type=1.2 Nibs=5 Dist=1419F
NUMC++ =  03690 JP%PR3          - 02B6A JP%PR1(005CA) Type=1.1 Nibs=4 Dist=00B26
NUMC+0 =  03696 JP%PR3          -
NUMCK  =  0369D JP%PR3          - 02A70 JP%PR1(004D0) Type=1.1 Nibs=4 Dist=00C2D
                                + 02AC8 JP%PR1(00528) Type=1.1 Nibs=4 Dist=00BD5
                                + 02B77 JP%PR1(005D7) Type=1.1 Nibs=4 Dist=00B26
                                + 033B4 JP%PR2(0078E) Type=1.1 Nibs=3 Dist=002E9
NUMCK0 =  03699 JP%PR3          - 032F9 JP%PR2(006D3) Type=1.1 Nibs=3 Dist=003A0
NUMSCN =  04D18 AB%LEX          -
NXPRMP =  0B3A5 AB%REG          -
NXTADR =  147E8 SC%DAT          - 17C1A SB%RD:(00053) Type=1.1 Nibs=4 Dist=03432
NXTCHR =  158B2 AB%CLC          - 16826 AB%ED:(001B4) Type=1.0 Nibs=4 Dist=00F74
NXTDC  =  0528A SG%LDC          - 08842 SG%EXC(001FA) Type=1.2 Nibs=5 Dist=035B8
NXTELM =  148AC SC%DAT          -
NXTEXP =  1C2F7 MB%USG          -
NXTIRQ =  2F70D SB%RAM          - 129B6 MN%TM:(0046C) Type=0.0 Nibs=5
                                + 129C1 MN%TM:(00477) Type=0.0 Nibs=5
                                + 13123 MN%TM:(00BD9) Type=0.0 Nibs=5
NXTLIN =  10031 JP%MEM          - 06D41 SG%SYS(009A1) Type=0.1 Nibs=5
                                + 06EAF SG%SYS(00B0F) Type=0.1 Nibs=5
NXTP   =  03455 JP%PR3          - 08847 SG%EXC(001FF) Type=1.2 Nibs=5 Dist=053F2
NXTST1 =  08A4B SG%EXC          - 07688 JP%SYS(00759) Type=1.0 Nibs=4 Dist=013C3
NXTST2 =  08A58 SG%EXC          - 1994F SC%SUB(00BDD) Type=0.1 Nibs=5
NXTSTM =  08A48 SG%EXC          - 0698C SG%SYS(005EC) Type=1.0 Nibs=4 Dist=020BC
                                + 076D7 JP%SYS(007A8) Type=1.0 Nibs=4 Dist=01371
                                + 08260 JP%EXC(00484) Type=1.0 Nibs=3 Dist=007E8
                                + 0A062 SG%FXQ(004EC) Type=1.0 Nibs=4 Dist=0161A
                                + 0B061 AB%REG(00469) Type=1.0 Nibs=4 Dist=02619
                                + 0E04B PM%STA(0021C) Type=1.0 Nibs=4 Dist=05603
                                + 0EDC8 MN%UTL(0015A) Type=1.0 Nibs=4 Dist=06380
                                + 0FA92 SC%TRC(0004B) Type=1.0 Nibs=4 Dist=0704A
                                + 18099 SB%IO:(002DD) Type=0.0 Nibs=5
                                + 1ED9D SB%TAB(00866) Type=1.2 Nibs=5 Dist=16355
                                + 1EDA6 SB%TAB(0086F) Type=1.2 Nibs=5 Dist=1635E
                                + 1EDC1 SB%TAB(0088A) Type=1.2 Nibs=5 Dist=16379
```

```
                                + 1EDD3 SB%TAB(0089C) Type=1.2 Nibs=5 Dist=1638B
                                + 1EE51 SB%TAB(0091A) Type=1.2 Nibs=5 Dist=16409
                                + 1EE5A SB%TAB(00923) Type=1.2 Nibs=5 Dist=16412
                                + 1EE87 SB%TAB(00950) Type=1.2 Nibs=5 Dist=1643F
NXTVA- =  13E58 SC%DAT          -
NXTVAR =  13E4C SC%DAT          - 17BF8 SB%RD:(00031) Type=1.1 Nibs=4 Dist=03DAC
NXtstn =  14384 SC%DAT          - 16B65 FH%TFM(000B7) Type=1.0 Nibs=4 Dist=027E1
                                + 17DB8 SB%RD:(001F1) Type=1.0 Nibs=4 Dist=03A34
                                + 188F0 SB%IO:(00B34) Type=1.0 Nibs=4 Dist=0456C
                                + 19A09 SC%SUB(00C97) Type=1.0 Nibs=4 Dist=05685
                                + 1A70E SC%REN(00192) Type=1.0 Nibs=4 Dist=0638A
NoCont =  0000E TI%EQU          - 00384 SB%DVR(00384) Type=0.0 Nibs=1
                                + 18B08 SB%IO:(00D4C) Type=0.0 Nibs=1
Ntoken =  02BC7 JP%PR1          - 02C30 JP%PR2(0000A) Type=1.1 Nibs=3 Dist=00069
                                + 03140 JP%PR2(0051A) Type=1.1 Nibs=3 Dist=00579
                                + 031D4 JP%PR2(005AE) Type=1.1 Nibs=3 Dist=0060D
                                + 032AB JP%PR2(00685) Type=1.1 Nibs=3 Dist=006E4
NwOFF+ =  1C026 MB%USG          -
NwOFFS =  1C02D MB%USG          -
Nxtf13 =  1B5D6 MB%IMG          - 1BC8A MB%USG(001E6) Type=1.0 Nibs=3 Dist=006B4
Nxtstn =  076D5 JP%SYS          - 00186 SB%DVR(00186) Type=1.0 Nibs=4 Dist=0754F

OAGNXT =  03060 JP%PR2          - 03826 JP%PR3(00450) Type=1.1 Nibs=3 Dist=007C6
OBCOLL =  01435 TI%UTL          - 02647 JP%PR1(000A7) Type=1.1 Nibs=4 Dist=01212
                                + 06C63 SG%SYS(008C3) Type=1.0 Nibs=4 Dist=0582E
                                + 0FF8B JP%MEM(0002C) Type=0.1 Nibs=5
                                + 1777A FH%TFM(00CCC) Type=0.1 Nibs=5
                                + 1D8EA MN%CD:(01071) Type=0.1 Nibs=5
OBEDIT =  17687 FH%TFM          -
OBLCMP =  0148C TI%UTL          - 0289C JP%PR1(002FC) Type=1.1 Nibs=4 Dist=01410
                                + 06C31 SG%SYS(00891) Type=1.1 Nibs=4 Dist=057A5
                                + 17268 FH%TFM(007BA) Type=0.1 Nibs=5
OBPRD  =  01481 TI%UTL          - 1726F FH%TFM(007C1) Type=0.1 Nibs=5
OFF    =  07F90 JP%EXC          - 1ED8B SB%TAB(00854) Type=1.2 Nibs=5 Dist=16DFB
OFFDC  =  0557D SG%LDC          - 07F86 JP%EXC(001AA) Type=1.2 Nibs=5 Dist=02A09
OFFFLG =  2F442 SB%RAM          -
OFFP   =  03365 JP%PR2          - 07F8B JP%EXC(001AF) Type=1.2 Nibs=5 Dist=04C26
OFFTMR =  07FBE JP%EXC          - 07B44 JP%SYS(00C15) Type=1.1 Nibs=3 Dist=0047A
OKP    =  00000 JT%MTH          -
ON     =  07E34 JP%EXC          - 1ED82 SB%TAB(0084B) Type=1.2 Nibs=5 Dist=16F4E
ONDC   =  054ED SG%LDC          - 07E2A JP%EXC(0004E) Type=1.2 Nibs=5 Dist=0293D
ONDC20 =  05501 SG%LDC          -
ONEDGT =  0F208 AB%EXP          - 1E752 SB%TAB(0021B) Type=1.2 Nibs=5 Dist=0F54A
                                + 1E75B SB%TAB(00224) Type=1.2 Nibs=5 Dist=0F553
                                + 1E764 SB%TAB(0022D) Type=1.2 Nibs=5 Dist=0F55C
                                + 1E76D SB%TAB(00236) Type=1.2 Nibs=5 Dist=0F565
                                + 1E776 SB%TAB(0023F) Type=1.2 Nibs=5 Dist=0F56E
                                + 1E77F SB%TAB(00248) Type=1.2 Nibs=5 Dist=0F577
                                + 1E788 SB%TAB(00251) Type=1.2 Nibs=5 Dist=0F580
                                + 1E791 SB%TAB(0025A) Type=1.2 Nibs=5 Dist=0F589
                                + 1E79A SB%TAB(00263) Type=1.2 Nibs=5 Dist=0F592
                                + 1E7A3 SB%TAB(0026C) Type=1.2 Nibs=5 Dist=0F59B
ONERR  =  07FF7 JP%EXC          - 09543 TI%ERD(001B0) Type=1.0 Nibs=4 Dist=0154C
ONINTR =  2F68D SB%RAM          - 19055 SC%SUB(002E3) Type=0.0 Nibs=5
ONP    =  02B28 JP%PR1          - 07E2F JP%EXC(00053) Type=1.2 Nibs=5 Dist=05307
ONP40  =  02B7B JP%PR1          -
ONTIMR =  08008 JP%EXC          - 075BA JP%SYS(0068B) Type=1.0 Nibs=4 Dist=00A4E
OPENF  =  11B06 SC%FIL          -
OPENF* =  11B11 SC%FIL          - 16FCD FH%TFM(0051F) Type=1.0 Nibs=4 Dist=054BC
OPENF- =  11B54 SC%FIL          -
```

```
OPNCHH  =  11AA9 SC%FIL        -
OPNF+   =  11B3F SC%FIL        -
OPSTRp  =  03784 JP%PR3        - 1A3DF SB%EXC(0000B) Type=1.2 Nibs=5 Dist=16C5B
OPTDC   =  05585 SG%LDC        - 137B4 PM%FLG(0036B) Type=1.2 Nibs=5 Dist=0E22F
OPTION  =  137BE PM%FLG        - 1EDF7 SB%TAB(008C0) Type=1.2 Nibs=5 Dist=0B639
OPTP    =  02D4E JP%PR2        - 137B9 PM%FLG(00370) Type=1.2 Nibs=5 Dist=10A6B
OR      =  0B891 AB%FCN        - 1EA97 SB%TAB(00560) Type=1.2 Nibs=5 Dist=13206
ORGSB   =  0D65B SM%MTH        -
ORIGIN  =  1586C AB%CLC        - 166FC AB%ED:(0008A) Type=1.1 Nibs=4 Dist=00E90
                               + 16849 AB%ED:(001D7) Type=1.1 Nibs=4 Dist=00FDD
ORSB    =  0D63C SM%MTH        - 0E1AB PM%STA(0037C) Type=1.1 Nibs=4 Dist=00B6F
ORXM    =  0D633 SM%MTH        -
OUT1T+  =  02CDF JP%PR2        - 0525A SG%LDC(002FC) Type=1.1 Nibs=4 Dist=0257B
OUT1TK  =  02CEB JP%PR2        - 0187C SB%CMD(00255) Type=1.1 Nibs=4 Dist=0146F
                               + 027BC JP%PR1(0021C) Type=1.1 Nibs=3 Dist=0052F
                               + 033FA JP%PR3(00024) Type=1.0 Nibs=3 Dist=0070F
                               + 0461E SB%EXP(00649) Type=1.0 Nibs=4 Dist=01933
                               + 05F94 SB%EXD(00672) Type=1.0 Nibs=4 Dist=032A9
                               + 17CD9 SB%RD:(00112) Type=0.1 Nibs=5
                               + 1882A SB%IO:(00A6E) Type=0.1 Nibs=5
OUT2TC  =  02CFD JP%PR2        - 02A27 JP%PR1(00487) Type=1.0 Nibs=3 Dist=002D6
                               + 0581C SG%LDC(008BE) Type=1.0 Nibs=4 Dist=02B1F
                               + 072B3 JP%SYS(00384) Type=1.1 Nibs=4 Dist=045B6
                               + 0FB25 SC%TRC(000DE) Type=0.1 Nibs=5
OUT2TK  =  02CFF JP%PR2        - 0268F JP%PR1(000EF) Type=1.1 Nibs=3 Dist=00670
                               + 02937 JP%PR1(00397) Type=1.1 Nibs=3 Dist=003C8
                               + 03E93 JP%PR3(00ABD) Type=1.0 Nibs=4 Dist=01194
                               + 05A59 SB%EXD(00137) Type=1.1 Nibs=4 Dist=02D5A
                               + 05A7B SB%EXD(00159) Type=1.1 Nibs=4 Dist=02D7C
OUT3TC  =  02D12 JP%PR2        - 058A6 SG%LDC(00948) Type=1.1 Nibs=4 Dist=02B94
OUT3TK  =  02D15 JP%PR2        - 027B5 JP%PR1(00215) Type=1.1 Nibs=3 Dist=00560
                               + 02A20 JP%PR1(00480) Type=1.1 Nibs=3 Dist=002F5
                               + 03BEF JP%PR3(00819) Type=1.0 Nibs=4 Dist=00EDA
OUT=1   =  06279 TI%FX1        - 005DB SB%DVR(005DB) Type=1.1 Nibs=4 Dist=05C9E
OUT=F   =  06281 TI%FX1        - 006CF SB%DVR(006CF) Type=1.1 Nibs=4 Dist=05BB2
OUTB+5  =  03836 JP%PR3        - 0272B JP%PR1(0018B) Type=1.1 Nibs=4 Dist=0110B
OUTBS   =  2F58F SB%RAM        - 01425 TI%UTL(002F4) Type=0.0 Nibs=5
                               + 01437 TI%UTL(00306) Type=0.0 Nibs=5
                               + 01483 TI%UTL(00352) Type=0.0 Nibs=5
                               + 028BF JP%PR1(0031F) Type=0.0 Nibs=5
                               + 05069 SG%LDC(0010B) Type=0.0 Nibs=5
                               + 06B2A SG%SYS(0078A) Type=0.0 Nibs=5
                               + 06D48 SG%SYS(009A8) Type=0.0 Nibs=5
OUTBY+  =  02CE5 JP%PR2        - 056C9 SG%LDC(0076B) Type=1.0 Nibs=4 Dist=029E4
OUTBYT  =  02CE8 JP%PR2        - 017A1 SB%CMD(0017A) Type=1.1 Nibs=4 Dist=01547
                               + 02814 JP%PR1(00274) Type=1.1 Nibs=3 Dist=004D4
                               + 0296B JP%PR1(003CB) Type=1.1 Nibs=3 Dist=0037D
                               + 02A15 JP%PR1(00475) Type=1.1 Nibs=3 Dist=002D3
                               + 02A34 JP%PR1(00494) Type=1.1 Nibs=3 Dist=002B4
                               + 039C0 JP%PR3(005EA) Type=1.0 Nibs=4 Dist=00CD8
                               + 05CC8 SB%EXD(003A6) Type=1.0 Nibs=4 Dist=02FE0
                               + 06C20 SG%SYS(00880) Type=1.1 Nibs=4 Dist=03F38
                               + 1ABFF SG%KEY(0015E) Type=0.1 Nibs=5
OUTC15  =  05421 SG%LDC        - 0A481 SG%FXQ(0090B) Type=1.1 Nibs=4 Dist=05060
                               + 1AACA SG%KEY(00029) Type=0.1 Nibs=5
OUTEL1  =  05300 SG%LDC        -
OUTELA  =  05303 SG%LDC        -
OUTLI1  =  03709 JP%PR3        - 04059 SB%EXP(00084) Type=1.1 Nibs=4 Dist=00950
OUTLIT  =  036F3 JP%PR3        -
OUTNB4  =  0383A JP%PR3        - 030B1 JP%PR2(0048B) Type=1.1 Nibs=3 Dist=00789
```

```
OUTNBC =  05423 SG%LDC        - 0176B SB%CMD(00144) Type=1.1 Nibs=4 Dist=03CB8
                              + 04297 SB%EXP(002C2) Type=1.1 Nibs=4 Dist=0118C
                              + 0444C SB%EXP(00477) Type=1.1 Nibs=4 Dist=00FD7
                              + 06043 SB%EXD(00721) Type=1.0 Nibs=4 Dist=00C20
OUTNBS =  05426 SG%LDC        - 02CB3 JP%PR2(0008D) Type=1.1 Nibs=4 Dist=02773
                              + 03CFA JP%PR3(00924) Type=1.0 Nibs=4 Dist=0172C
                              + 0467A SB%EXP(006A5) Type=1.0 Nibs=4 Dist=00DAC
                              + 05BB4 SB%EXD(00292) Type=1.1 Nibs=3 Dist=0078E
                              + 05DED SB%EXD(004CB) Type=1.1 Nibs=4 Dist=009C7
                              + 06758 SG%SYS(003B8) Type=1.1 Nibs=4 Dist=01332
                              + 068FB SG%SYS(0055B) Type=1.1 Nibs=4 Dist=014D5
                              + 0A489 SG%FXQ(00913) Type=1.1 Nibs=4 Dist=05063
                              + 1AB98 SG%KEY(000F7) Type=0.1 Nibs=5
OUTNIB =  02D28 JP%PR2        - 03803 JP%PR3(0042D) Type=1.1 Nibs=4 Dist=00ADB
                              + 041AD SB%EXP(001D8) Type=1.1 Nibs=4 Dist=01485
OUTOVF =  02D48 JP%PR2        - 028B2 JP%PR1(00312) Type=1.0 Nibs=3 Dist=00496
OUTRES =  0BC84 AB%FCN        -
OUTRP  =  05CC2 SB%EXD        - 056D6 SG%LDC(00778) Type=1.0 Nibs=3 Dist=005EC
OUTVAR =  0373E JP%PR3        - 031B1 JP%PR2(0058B) Type=1.1 Nibs=3 Dist=0058D
                              + 037D5 JP%PR3(003FF) Type=1.1 Nibs=3 Dist=00097
                              + 03811 JP%PR3(0043B) Type=1.1 Nibs=3 Dist=000D3
                              + 04025 SB%EXP(00050) Type=1.1 Nibs=4 Dist=008E7
                              + 04046 SB%EXP(00071) Type=1.1 Nibs=4 Dist=00908
                              + 041A4 SB%EXP(001CF) Type=1.1 Nibs=4 Dist=00A66
                              + 0437A SB%EXP(003A5) Type=1.1 Nibs=4 Dist=00C3C
                              + 048B1 SB%EXP(008DC) Type=1.1 Nibs=4 Dist=01173
OUTbyt =  039BE JP%PR3        - 03FD6 SB%EXP(00001) Type=1.0 Nibs=3 Dist=00618
OUtbyt =  1ABFD SG%KEY        - 1ADE3 SB%VAL(00075) Type=1.1 Nibs=3 Dist=001E6
OVERCK =  1723B FH%TFM        -
OVF    =  13893 PM%FLG        - 1EBC9 SB%TAB(00692) Type=1.2 Nibs=5 Dist=0B336
OVF-15 =  0D1F7 JT%MTH        -
OVF15? =  0D1E1 JT%MTH        -
OVFL   =  0CA73 JT%MTH        - 0F9DD AB%ASN(00403) Type=1.1 Nibs=4 Dist=02F6A
OVFNIB =  2F6FB SB%RAM        - 046C1 SB%EXP(006EC) Type=0.0 Nibs=5
                              + 161B9 AB%CLC(00C9A) Type=0.0 Nibs=5
OVP    =  00002 JT%MTH        - 046FA SB%EXP(00725) Type=0.0 Nibs=1

P1-10  =  041C1 SB%EXP        -
PARERR =  02F08 JP%PR2        -
PARPRP =  160E8 AB%CLC        - 16793 AB%ED:(00121) Type=1.1 Nibs=3 Dist=006AB
PART3  =  18097 SB%IO:        - 1C147 MB%USG(006A3) Type=1.0 Nibs=4 Dist=040B0
PAUSE  =  076D2 JP%SYS        - 1ED31 SB%TAB(007FA) Type=1.2 Nibs=5 Dist=1765F
PCADDR =  2F679 SB%RAM        - 07485 JP%SYS(00556) Type=0.0 Nibs=5
                              + 07B67 JP%SYS(00C38) Type=0.0 Nibs=5
                              + 088FE SG%EXC(002B6) Type=0.0 Nibs=5
                              + 08A4D SG%EXC(00405) Type=0.0 Nibs=5
                              + 09B39 TI%ERD(007A6) Type=0.0 Nibs=5
                              + 1463E SC%DAT(00D89) Type=0.0 Nibs=5
                              + 18B22 SB%IO:(00D66) Type=0.0 Nibs=5
                              + 19783 SC%SUB(00A11) Type=0.0 Nibs=5
                              + 1A5A9 SC%REN(0002D) Type=0.0 Nibs=5
PCEXPR =  06828 SG%SYS        - 1C573 SG%POK(00085) Type=0.1 Nibs=5
PDEV   =  09E9E SG%FXQ        -
PDEV+  =  09E9B SG%FXQ        - 17A3A MN%FRP(00007) Type=0.1 Nibs=5
                              + 17B35 MN%FRP(00102) Type=0.1 Nibs=5
PDEV1  =  09EA4 SG%FXQ        - 0BF99 AB%FCN(00960) Type=1.1 Nibs=4 Dist=020F5
PEDIT  =  0FF5F JP%MEM        - 0288F JP%PR1(002EF) Type=0.1 Nibs=5
PEDITD =  0FF62 JP%MEM        - 026BC JP%PR1(0011C) Type=0.1 Nibs=5
PEDITM =  0FF70 JP%MEM        - 06D6F SG%SYS(009CF) Type=0.1 Nibs=5
PEEK$  =  1C50D SG%POK        - 1DF9D JP%TAB(00299) Type=1.2 Nibs=5 Dist=01A90
```

```
PERCNT =  0B698 AB%FCN        - 1EA4F SB%TAB(00518) Type=1.2 Nibs=5 Dist=133B7
PFINDL =  078DF JP%SYS        - 1B468 MB%IMG(00022) Type=0.1 Nibs=5
PFNDL* =  07932 JP%SYS        - 1A963 SC%REN(003E7) Type=0.1 Nibs=5
PFNDZL =  078E2 JP%SYS        -
PI     =  0C000 AB%FCN        - 1E9E3 SB%TAB(004AC) Type=1.2 Nibs=5 Dist=129E3
PI/180 =  0C196 AB%FCN        -
PI/2   =  0DB77 SM%MTH        -
PI/2D  =  0DB7A SM%MTH        -
PI/4   =  0DAA1 SM%MTH        -
PILP   =  033D6 JP%PR3        - 03238 JP%PR2(00612) Type=1.1 Nibs=3 Dist=0019E
PILP+  =  033D9 JP%PR3        - 02A62 JP%PR1(004C2) Type=1.1 Nibs=4 Dist=00977
PKFIX1 =  1DBAA TI%FX4        - 1C6B2 SG%POK(001C4) Type=1.1 Nibs=4 Dist=014F8
PLIST  =  06A40 SG%SYS        - 1DE50 JP%TAB(0014C) Type=1.2 Nibs=5 Dist=17410
PLUS   =  0B6BC AB%FCN        - 1EA61 SB%TAB(0052A) Type=1.2 Nibs=5 Dist=133A5
PNDALM =  2F761 SB%RAM        - 074CC JP%SYS(0059D) Type=0.0 Nibs=4
                              + 07551 JP%SYS(00622) Type=0.0 Nibs=5
                              + 07588 JP%SYS(00659) Type=0.0 Nibs=5
                              + 12A3F MN%TM:(004F5) Type=0.0 Nibs=5
                              + 12A4B MN%TM:(00501) Type=0.0 Nibs=5
POKE   =  1C629 SG%POK        - 1DFA6 JP%TAB(002A2) Type=1.2 Nibs=5 Dist=0197D
POKE25 =  1C699 SG%POK        - 1C4BD SB%GPH(000F8) Type=1.0 Nibs=3 Dist=001DC
POKEP  =  0378E JP%PR3        - 1C624 SG%POK(00136) Type=1.2 Nibs=5 Dist=18E96
POKERR =  1C776 SG%POK        - 1DBCE TI%FX4(00024) Type=1.0 Nibs=4 Dist=01458
POLL   =  12337 JP%POL        - 0A455 SG%FXQ(008DF) Type=1.0 Nibs=4 Dist=07EE2
                              + 116BD SC%FIL(0066A) Type=1.0 Nibs=4 Dist=00C7A
                              + 13E2E SC%DAT(00579) Type=1.0 Nibs=4 Dist=01AF7
                              + 17376 FH%TFM(008C8) Type=1.1 Nibs=4 Dist=0503F
                              + 18599 SB%IO:(007DD) Type=1.1 Nibs=4 Dist=06262
                              + 185C8 SB%IO:(0080C) Type=1.1 Nibs=4 Dist=06291
                              + 19082 SC%SUB(00310) Type=1.1 Nibs=4 Dist=06D4B
                              + 196FD SC%SUB(0098B) Type=1.1 Nibs=4 Dist=073C6
                              + 1C9A2 MN%CD:(00129) Type=0.1 Nibs=5
POLLD+ =  1232D JP%POL        - 03D95 JP%PR3(009BF) Type=0.1 Nibs=5
POP    =  08FCF SG%EXC        - 1DFAF JP%TAB(002AB) Type=1.2 Nibs=5 Dist=14FE0
POP1N  =  0BD1C AB%FCN        - 0AD6D AB%REG(00175) Type=1.1 Nibs=4 Dist=00FAF
                              + 0E8FF PM%STA(00AD0) Type=1.1 Nibs=4 Dist=02BE3
                              + 11190 SC%FIL(0013D) Type=1.1 Nibs=4 Dist=05474
                              + 15DDB AB%CLC(008BC) Type=0.1 Nibs=5
                              + 15E53 AB%CLC(00934) Type=0.1 Nibs=5
                              + 1656F AB%BLD(00243) Type=0.1 Nibs=5
                              + 17FF7 SB%IO:(0023B) Type=0.1 Nibs=5
                              + 18117 SB%IO:(0035B) Type=0.1 Nibs=5
                              + 18162 SB%IO:(003A6) Type=0.1 Nibs=5
                              + 1AEB7 SB%VAL(00149) Type=0.1 Nibs=5
POP1N+ =  0BD91 AB%FCN        - 09189 SG%EXC(00B41) Type=1.1 Nibs=4 Dist=02C08
                              + 0E672 PM%STA(00843) Type=1.1 Nibs=4 Dist=028E1
                              + 13520 PM%FLG(000D7) Type=1.1 Nibs=4 Dist=0778F
POP1R  =  0E8FD PM%STA        - 08878 SG%EXC(00230) Type=1.1 Nibs=4 Dist=06085
                              + 089DE SG%EXC(00396) Type=1.1 Nibs=4 Dist=05F1F
                              + 1C4F0 SG%POK(00002) Type=0.1 Nibs=5
POP1S  =  0BD38 AB%FCN        - 08D9D SG%EXC(00755) Type=1.1 Nibs=4 Dist=02F9B
                              + 0A9C0 SB%FCN(00095) Type=1.1 Nibs=4 Dist=01378
                              + 0AA57 SB%FCN(0012C) Type=1.1 Nibs=4 Dist=012E1
                              + 12EF6 MN%TM:(009AC) Type=1.1 Nibs=4 Dist=071BE
                              + 18ACD SB%IO:(00D11) Type=0.1 Nibs=5
                              + 1A40C SB%EXC(00038) Type=0.1 Nibs=5
                              + 1A450 SB%EXC(0007C) Type=0.1 Nibs=5
                              + 1B390 AB%UTL(00371) Type=0.1 Nibs=5
                              + 1C468 SB%GPH(000A3) Type=0.1 Nibs=5
                              + 1C649 SG%POK(0015B) Type=0.1 Nibs=5
```

```
POP2N  =  0BC8C AB%FCN        -
POP2N+ =  0BD58 AB%FCN        - 0E8F9 PM%STA(00ACA) Type=1.0 Nibs=4 Dist=02BA1
POPARG =  1118E SC%FIL        - 19287 SC%SUB(00515) Type=0.1 Nibs=5
POPBUF =  010EE SB%KEY        - 00165 SB%DVR(00165) Type=0.1 Nibs=5
                              + 021B0 SB%DSP(00920) Type=1.1 Nibs=4 Dist=010C2
                              + 021E0 SB%DSP(00950) Type=1.1 Nibs=4 Dist=010F2
                              + 021EA SB%DSP(0095A) Type=1.1 Nibs=4 Dist=010FC
                              + 0658B SG%SYS(001EB) Type=1.0 Nibs=4 Dist=0549D
                              + 14E80 MN%ED:(00200) Type=0.1 Nibs=5
                              + 1ACBF SG%KEY(0021E) Type=0.1 Nibs=5
POPCH# =  12267 SC%FIL        - 1977D SC%SUB(00A0B) Type=1.0 Nibs=4 Dist=07516
POPGSB =  08F3E SG%EXC        - 07B13 JP%SYS(00BE4) Type=1.1 Nibs=4 Dist=0142B
                              + 1AF38 MN%GSB(00037) Type=0.1 Nibs=5
POPMTH =  1B3DB AB%UTL        - 0928B SG%EXC(00C43) Type=0.1 Nibs=5
                              + 0E4A2 PM%STA(00673) Type=0.1 Nibs=5
                              + 0FDEF SC%TRC(003A8) Type=0.1 Nibs=5
                              + 13D45 SC%DAT(00490) Type=1.1 Nibs=4 Dist=07696
                              + 13F56 SC%DAT(006A1) Type=1.1 Nibs=4 Dist=07485
                              + 18530 SB%IO:(00774) Type=1.0 Nibs=4 Dist=02EAB
                              + 1C0A4 MB%USG(00600) Type=1.1 Nibs=4 Dist=00CC9
POPSTK =  08F55 SG%EXC        - 06DA0 SG%SYS(00A00) Type=1.1 Nibs=4 Dist=021B5
                              + 1AEA7 SB%VAL(00139) Type=0.1 Nibs=5
POPSTR =  1B405 AB%UTL        - 18F1B SC%SUB(001A9) Type=1.1 Nibs=4 Dist=024EA
POPUPD =  08F3E SG%EXC        - 124CD JP%POL(001B2) Type=0.1 Nibs=5
                              + 16CB6 FH%TFM(00208) Type=0.1 Nibs=5
                              + 16F1E FH%TFM(00470) Type=0.1 Nibs=5
POS    =  0C01B AB%FCN        - 1DFB8 JP%TAB(002B4) Type=1.2 Nibs=5 Dist=11F9D
POllj  =  0A453 SG%FXQ        - 06B83 SG%SYS(007E3) Type=1.0 Nibs=4 Dist=038D0
                              + 076EA JP%SYS(007BB) Type=1.0 Nibs=4 Dist=02D69
                              + 08453 JP%EXC(00677) Type=1.0 Nibs=4 Dist=02000
                              + 093B8 TI%ERD(00025) Type=1.0 Nibs=4 Dist=0109B
POlljj =  13E2C SC%DAT        - 1A876 SC%REN(002FA) Type=1.1 Nibs=4 Dist=06A4A
PPOS   =  2F956 SB%RAM        - 185BA SB%IO:(007FE) Type=0.0 Nibs=5
PREDV  =  0E31D PM%STA        - 1EB39 SB%TAB(00602) Type=1.2 Nibs=5 Dist=1081C
PRENCK =  03681 JP%PR3        - 03101 JP%PR2(004DB) Type=1.1 Nibs=3 Dist=00580
                              + 033B8 JP%PR2(00792) Type=1.1 Nibs=3 Dist=002C9
PREP   =  0ADAF AB%REG        -
PRESCN =  04A49 AB%LEX        - 040C3 SB%EXP(000EE) Type=1.1 Nibs=4 Dist=00986
PRGFE  =  0A118 SG%FXQ        - 01337 TI%UTL(00206) Type=0.1 Nibs=5
                              + 085D2 JP%EXC(007F6) Type=1.0 Nibs=4 Dist=01B46
PRGFMF =  0A146 SG%FXQ        - 07016 JP%SYS(000E7) Type=1.1 Nibs=4 Dist=03130
                              + 1736F FH%TFM(008C1) Type=0.1 Nibs=5
PRGMEN =  2F567 SB%RAM        - 0725C JP%SYS(0032D) Type=0.0 Nibs=5
                              + 07469 JP%SYS(0053A) Type=0.0 Nibs=5
                              + 0791D JP%SYS(009EE) Type=0.0 Nibs=5
                              + 08506 JP%EXC(0072A) Type=0.0 Nibs=5
                              + 087A1 SG%EXC(00159) Type=0.0 Nibs=5
                              + 17C28 SB%RD:(00061) Type=0.0 Nibs=5
                              + 19064 SC%SUB(002F2) Type=0.0 Nibs=5
PRGMST =  2F562 SB%RAM        - 07788 JP%SYS(00859) Type=0.0 Nibs=5
                              + 07BCD JP%SYS(00C9E) Type=0.0 Nibs=5
                              + 0915D SG%EXC(00B15) Type=0.0 Nibs=5
                              + 17C42 SB%RD:(0007B) Type=0.0 Nibs=4
                              + 19502 SC%SUB(00790) Type=0.0 Nibs=5
                              + 1A0AA SC%SUB(01338) Type=0.0 Nibs=5
                              + 1A21E SC%SUB(014AC) Type=0.0 Nibs=5
PRGROM =  0A30E SG%FXQ        - 07C59 JP%SYS(00D2A) Type=1.0 Nibs=4 Dist=026B5
PRINT  =  17F0E SB%IO:        - 1ECD7 SB%TAB(007A0) Type=1.2 Nibs=5 Dist=06DC9
PRINT# =  14291 SC%DAT        - 17F25 SB%IO:(00169) Type=1.0 Nibs=4 Dist=03C94
PRINT* =  17F37 SB%IO:        -
```

```
PRINTt =  00001 TIXEQU        -
PRIVAT =  0A440 SGXFXQ        - 1DFC1 JPXTAB(002BD) Type=1.2 Nibs=5 Dist=13B81
PRIVTP =  03B82 JPXPR3        - 0A43B SGXFXQ(008C5) Type=1.2 Nibs=5 Dist=068B9
PRMC10 =  0B37F ABXREG        -
PRMCHN =  0B375 ABXREG        - 19EE7 SCXSUB(01175) Type=0.1 Nibs=5
PRMCNT =  2F94B SBXRAM        - 038FF JPXPR3(00529) Type=0.0 Nibs=5
                              + 0482E SBXEXP(00859) Type=0.0 Nibs=5
                              + 18D85 SCXSUB(00013) Type=0.0 Nibs=5
                              + 19666 SCXSUB(008F4) Type=0.0 Nibs=5
                              + 1A290 SCXSUB(0151E) Type=0.0 Nibs=5
PRMPTR =  2F5B7 SBXRAM        - 0764D JPXSYS(0071E) Type=0.0 Nibs=5
                              + 07D67 JPXSYS(00E38) Type=0.0 Nibs=4
                              + 0B3C0 ABXREG(007C8) Type=0.0 Nibs=5
                              + 0F530 ABXEXP(003B8) Type=0.0 Nibs=5
                              + 10ABF MNXCNF(0092D) Type=0.0 Nibs=4 Offset=      2
                              + 1996B SCXSUB(00BF9) Type=0.0 Nibs=5
                              + 19D9E SCXSUB(0102C) Type=0.0 Nibs=5
                              + 1A078 SCXSUB(01306) Type=0.0 Nibs=5
                              + 1A12E SCXSUB(013BC) Type=0.0 Nibs=5 Offset=      2
                              + 1A158 SCXSUB(013E6) Type=0.0 Nibs=5
PRNEXe =  02E95 JPXPR2        -
PRNTDC =  05450 SGXLDC        - 17F04 SBXIO:(00148) Type=1.2 Nibs=5 Dist=12AB4
PROMPT =  0032C SBXDVR        - 01BF8 SBXDSP(00368) Type=0.0 Nibs=5
PROTCT =  1D715 MNXCD:        - 1DFCA JPXTAB(002C6) Type=1.2 Nibs=5 Dist=008B5
PRP3>3 =  09684 TIXERD        - 0965C TIXERD(002C9) Type=0.0 Nibs=5
PRPSND =  06B17 SGXSYS        - 1AD0D SGXKEY(0026C) Type=0.1 Nibs=5
PRSC00 =  07B93 JPXSYS        - 18FE9 SCXSUB(00277) Type=0.1 Nibs=5
PRSC20 =  07BCB JPXSYS        -
PRSC60 =  07BD4 JPXSYS        - 1A225 SCXSUB(014B3) Type=0.1 Nibs=5
PRSCKB =  07B88 JPXSYS        - 17BEB SBXRD:(00024) Type=0.1 Nibs=5
                              + 19A4F SCXSUB(00CDD) Type=0.1 Nibs=5
                              + 1A0A3 SCXSUB(01331) Type=0.1 Nibs=5
                              + 1B45D MBXIMG(00017) Type=0.1 Nibs=5
PRSCO- =  07B80 JPXSYS        - 1A2D5 SCXSUB(01563) Type=0.1 Nibs=5
PRSCOP =  07B8F JPXSYS        - 1A595 SCXREN(00019) Type=0.1 Nibs=5
PRSsc+ =  1BA84 MBXIMG        -
PRSscn =  1BA88 MBXIMG        -
PRT#DC =  06841 SGXSYS        - 0A499 SGXFXQ(00923) Type=1.1 Nibs=4 Dist=03C58
PRTP   =  03599 JPXPR3        - 17F09 SBXIO:(0014D) Type=1.2 Nibs=5 Dist=14970
PSHGSB =  08F13 SGXEXC        - 07A81 JPXSYS(00B52) Type=1.1 Nibs=4 Dist=01492
PSHMCR =  08F0B SGXEXC        - 1AF14 MNXGSB(00013) Type=0.1 Nibs=5
PSHSTK =  08C7F SGXEXC        - 1AE1F SBXVAL(000B1) Type=0.1 Nibs=5
PSHSTL =  08C85 SGXEXC        - 123D3 JPXPOL(000B8) Type=0.1 Nibs=5
PSHSTM =  15582 ABXCLC        - 168AB ABXED:(00239) Type=1.1 Nibs=4 Dist=01329
PSHUPD =  08F0D SGXEXC        - 06D03 SGXSYS(00963) Type=1.0 Nibs=4 Dist=0220A
                              + 07A64 JPXSYS(00B35) Type=1.1 Nibs=4 Dist=014A9
                              + 16BA8 FHXTFM(000FA) Type=0.1 Nibs=5
PTRAD1 =  08F83 SGXEXC        -
PTRAD2 =  08F8A SGXEXC        - 11A96 SCXFIL(00A43) Type=0.1 Nibs=5
PTRRCL =  0F3A7 ABXEXP        - 199F0 SCXSUB(00C7E) Type=0.1 Nibs=5
                              + 19E30 SCXSUB(010BE) Type=0.1 Nibs=5
PUGFIB =  12198 SCXFIL        - 0A204 SGXFXQ(0068E) Type=1.1 Nibs=4 Dist=07F94
                              + 170FB FHXTFM(0064D) Type=1.0 Nibs=4 Dist=04F63
PURGDC =  05745 SGXLDC        - 06422 SGXSYS(00082) Type=1.2 Nibs=5 Dist=00CDD
                              + 0A03F SGXFXQ(004C9) Type=1.2 Nibs=5 Dist=048FA
PURGE  =  0A049 SGXFXQ        - 1EDE5 SBXTAB(008AE) Type=1.2 Nibs=5 Dist=14D9C
PURGEF =  17359 FHXTFM        -
PURGEP =  03A89 JPXPR3        - 0A044 SGXFXQ(004CE) Type=1.2 Nibs=5 Dist=065BB
PUSH-3 =  045DE SBXEXP        -
PUSH-P =  045E0 SBXEXP        -
```

```
PUT     =  092C1 SG%EXC        - 1DFD3 JP%TAB(002CF) Type=1.2 Nibs=5 Dist=14D12
PUTALM  =  12952 MN%TM:        - 07D7E JP%SYS(00E4F) Type=0.1 Nibs=5
                               + 07FCC JP%EXC(001F0) Type=0.1 Nibs=5
PUTPND  =  12A49 MN%TM:        - 00753 SB%DVR(00753) Type=0.1 Nibs=5
PUTRES  =  18115 SB%IO:        - 1C0C2 MB%USG(0061E) Type=1.1 Nibs=4 Dist=03FAD
PWCONF  =  1020F MN%CNF        - 001D8 SB%DVR(001D8) Type=0.1 Nibs=5
                               + 005E4 SB%DVR(005E4) Type=0.1 Nibs=5
PWIDTH  =  2F958 SB%RAM        - 0EFC3 MN%UTL(00355) Type=0.0 Nibs=2
PWIDTX  =  0EFB9 MN%UTL        - 1DFDC JP%TAB(002D8) Type=1.2 Nibs=5 Dist=0F023
PWROFF  =  00526 SB%DVR        - 07FE9 JP%EXC(0020D) Type=1.0 Nibs=4 Dist=07AC3
PWROFS  =  0055F SB%DVR        - 00584 SB%DVR(00584) Type=0.0 Nibs=5
PgmRun  =  0000D TI%EQU        - 02155 SB%DSP(008C5) Type=0.0 Nibs=1
Polrtn  =  1B986 MB%IMG        - 1BB96 MB%USG(000F2) Type=1.1 Nibs=3 Dist=00210
                               + 1BD08 MB%USG(00264) Type=1.1 Nibs=3 Dist=00382

QUOEXe  =  02E8B JP%PR2        -
QUOTCK  =  0623D SB%EXD        - 04050 SB%EXP(0007B) Type=1.1 Nibs=4 Dist=021ED
                               + 06292 TI%FX1(00042) Type=1.1 Nibs=3 Dist=00055

R.STPR  =  029CD JP%PR1        - 02FB3 JP%PR2(0038D) Type=1.1 Nibs=3 Dist=005E6
                               + 034E0 JP%PR3(0010A) Type=1.1 Nibs=4 Dist=00B13
R1REV   =  00785 Define        - 0A95B SB%FCN(00030) Type=0.0 Nibs=5
R2REV   =  0AA83 Define        - 0A966 SB%FCN(0003B) Type=0.0 Nibs=4
R3=D1*  =  03522 JP%PR3        - 02AF3 JP%PR1(00553) Type=1.1 Nibs=4 Dist=00A2F
                               + 0313C JP%PR2(00516) Type=1.1 Nibs=3 Dist=003E6
R3=D1+  =  03532 JP%PR3        - 1BC75 MB%USG(001D1) Type=0.1 Nibs=5
R3=D10  =  03526 JP%PR3        - 051ED SG%LDC(0028F) Type=1.1 Nibs=4 Dist=01CC7
                               + 097FE TI%ERD(0046B) Type=1.1 Nibs=4 Dist=062D8
                               + 1B52C MB%IMG(000E6) Type=0.1 Nibs=5
R3=D1C  =  0352C JP%PR3        -
R3REV   =  153AB Define        - 0A970 SB%FCN(00045) Type=0.0 Nibs=5
R4REV   =  1DBA8 Define        - 0A97B SB%FCN(00050) Type=0.0 Nibs=4
R<RST2  =  014DB TI%UTL        - 04740 SB%EXP(0076B) Type=1.1 Nibs=4 Dist=03265
                               + 094AB TI%ERD(00118) Type=0.1 Nibs=5
                               + 1CF33 MN%CD:(006BA) Type=0.1 Nibs=5
R<RSTK  =  014DD TI%UTL        - 01C41 SB%DSP(003B1) Type=1.1 Nibs=3 Dist=00764
                               + 12464 JP%POL(00149) Type=0.1 Nibs=5
R<Rstk  =  12462 JP%POL        - 109E6 MN%CNF(00854) Type=1.0 Nibs=4 Dist=01A7C
                               + 17296 FH%TFM(007E8) Type=1.0 Nibs=4 Dist=04E34
R<rstk  =  17292 FH%TFM        - 16BA3 FH%TFM(000F5) Type=1.1 Nibs=3 Dist=006EF
RACTMI  =  08073 JP%EXC        - 0909D SG%EXC(00A55) Type=1.1 Nibs=4 Dist=0102A
RAD     =  0C181 AB%FCN        - 1E980 SB%TAB(00449) Type=1.2 Nibs=5 Dist=127FF
RADIAN  =  13875 PM%FLG        - 1ED16 SB%TAB(007DF) Type=1.2 Nibs=5 Dist=0B4A1
RAMEND  =  2F582 SB%RAM        - 0A798 SG%FXQ(00C22) Type=0.0 Nibs=5
                               + 1048B MN%CNF(002F9) Type=0.0 Nibs=5
                               + 19766 SC%SUB(009F4) Type=0.0 Nibs=5
RAMROM  =  0A5F7 SG%FXQ        - 08574 JP%EXC(00798) Type=1.1 Nibs=4 Dist=02083
RANDOM  =  0E656 PM%STA        - 1DFE5 JP%TAB(002E1) Type=1.2 Nibs=5 Dist=0F98F
RANDP   =  03A6E JP%PR3        - 0E651 PM%STA(00822) Type=1.2 Nibs=5 Dist=0ABE3
RANGE   =  1B07C AB%UTL        - 045D9 SB%EXP(00604) Type=0.1 Nibs=5
                               + 15105 MN%ED:(00485) Type=1.0 Nibs=4 Dist=05F77
                               + 15F44 AB%CLC(00A25) Type=1.0 Nibs=4 Dist=05138
                               + 1AC39 SG%KEY(00198) Type=1.1 Nibs=3 Dist=00443
                               + 1C608 SG%POK(0011A) Type=1.1 Nibs=4 Dist=0158C
RAWBFR  =  2F580 SB%RAM        - 016A3 SB%CMD(0007C) Type=0.0 Nibs=4
                               + 01753 SB%CMD(0012C) Type=0.0 Nibs=5
                               + 017AA SB%CMD(00183) Type=0.0 Nibs=5
                               + 017CC SB%CMD(001A5) Type=0.0 Nibs=5
                               + 028D6 JP%PR1(00336) Type=0.0 Nibs=5
                               + 155FD AB%CLC(000DE) Type=0.0 Nibs=4
```

```
                                + 15736 AB%CLC(00217) Type=0.0 Nibs=4
                                + 15865 AB%CLC(00346) Type=0.0 Nibs=5
                                + 15992 AB%CLC(00473) Type=0.0 Nibs=4
RBLDCH =  19795 SC%SUB          - 07DD7 JP%SYS(00EA8) Type=0.1 Nibs=5
RC01   =  12AC6 MN%TM:          - 08536 JP%EXC(0075A) Type=0.1 Nibs=5
                                + 1C932 MN%CD:(000B9) Type=0.1 Nibs=5
RCCD1  =  0D3F5 JT%MTH          - 0DB74 SM%MTH(0073F) Type=1.0 Nibs=3 Dist=0077F
RCCD1X =  0DB73 SM%MTH          - 0DEC8 PM%STA(00099) Type=1.1 Nibs=3 Dist=00355
                                + 0E01A PM%STA(001EB) Type=1.1 Nibs=3 Dist=004A7
                                + 0E1E7 PM%STA(003B8) Type=1.1 Nibs=3 Dist=00674
RCCD2  =  0D41C JT%MTH          - 0DCC2 SM%MTH(0088D) Type=1.1 Nibs=4 Dist=008A6
RCCRLF =  0228C SB%DSP          - 022B0 SB%DSP(00A20) Type=0.0 Nibs=5
RCKBp  =  0254A MB%ROM          - 00FF9 SB%KEY(00302) Type=1.1 Nibs=4 Dist=01551
RCL*   =  0E983 PM%STA          -
RCLNUM =  14948 SC%DAT          - 0FDB0 SC%TRC(00369) Type=1.1 Nibs=4 Dist=04B98
RCLSTA =  01BA0 SB%DSP          - 0058B SB%DVR(0058B) Type=1.1 Nibs=4 Dist=01615
                                + 1516F MN%ED:(004EF) Type=0.1 Nibs=5
                                + 15213 MN%ED:(00593) Type=0.1 Nibs=5
                                + 15257 MN%ED:(005D7) Type=0.1 Nibs=5
                                + 1C4A0 SB%GPH(000DB) Type=0.1 Nibs=5
RCLW1  =  0E981 PM%STA          -
RCLW2  =  0E9BE PM%STA          -
RCLW3  =  0E9C4 PM%STA          -
RCSCR  =  0E954 PM%STA          -
RCTTM1 =  131B2 MN%TM:          - 08095 JP%EXC(002B9) Type=0.1 Nibs=5
RCVOFS =  1C050 MB%USG          -
RDATTY =  17CC6 SB%RD:          - 08AE7 SG%EXC(0049F) Type=0.1 Nibs=5
                                + 0BD17 AB%FCN(006DE) Type=0.1 Nibs=5
                                + 111A8 SC%FIL(00155) Type=1.0 Nibs=4 Dist=06B1E
                                + 14429 SC%DAT(00B74) Type=1.0 Nibs=4 Dist=0389D
                                + 16328 AB%CLC(00E09) Type=1.0 Nibs=4 Dist=0199E
                                + 18F8C SC%SUB(0021A) Type=1.0 Nibs=4 Dist=012C6
                                + 1AD75 SB%VAL(00007) Type=1.0 Nibs=4 Dist=030AF
RDBAS  =  173FF FH%TFM          -
RDBYTA =  13A2F SC%DAT          - 17620 FH%TFM(00B72) Type=1.1 Nibs=4 Dist=03BF1
RDCHD+ =  076EE JP%SYS          -
RDCHDR =  076F0 JP%SYS          - 100F2 JP%MEM(00193) Type=0.1 Nibs=5
RDENTY =  06A25 SG%SYS          - 0A3EA SG%FXQ(00874) Type=1.1 Nibs=4 Dist=039C5
RDHDR1 =  076FD JP%SYS          - 08381 JP%EXC(005A5) Type=1.1 Nibs=4 Dist=00C84
                                + 10F2E MN%CNF(00D9C) Type=0.1 Nibs=5
RDINFD =  08468 JP%EXC          - 011EE TI%UTL(000BD) Type=1.1 Nibs=4 Dist=0727A
RDINFO =  0846B JP%EXC          - 012E1 TI%UTL(001B0) Type=1.1 Nibs=4 Dist=0718A
                                + 0A276 SG%FXQ(00700) Type=1.1 Nibs=4 Dist=01E0B
                                + 0A2A7 SG%FXQ(00731) Type=1.1 Nibs=4 Dist=01E3C
                                + 17285 FH%TFM(007D7) Type=0.1 Nibs=5
RDINFS =  08461 JP%EXC          - 011D6 TI%UTL(000A5) Type=1.1 Nibs=4 Dist=0728B
                                + 07059 JP%SYS(0012A) Type=1.1 Nibs=4 Dist=01408
RDIZDC =  053F4 SG%LDC          - 0E64C PM%STA(0081D) Type=1.2 Nibs=5 Dist=09258
RDLNAS =  13A1F SC%DAT          - 1131D SC%FIL(002CA) Type=1.1 Nibs=4 Dist=02702
RDLNFX =  13A07 SC%DAT          -
RDTEXT =  17489 FH%TFM          -
READ   =  17BD1 SB%RD:          - 1ECA1 SB%TAB(0076A) Type=1.2 Nibs=5 Dist=070D0
READ#  =  13EBB SC%DAT          - 17BDF SB%RD:(00018) Type=1.0 Nibs=4 Dist=03D24
READ95 =  17DB6 SB%RD:          - 17B24 MN%FRP(000F1) Type=1.0 Nibs=3 Dist=00292
READDC =  0548A SG%LDC          - 17BC7 SB%RD:(00000) Type=1.2 Nibs=5 Dist=1273D
READIN =  0F484 AB%EXP          - 14972 SC%DAT(010BD) Type=1.1 Nibs=4 Dist=054EE
READNB =  17518 FH%TFM          -
READP  =  03237 JP%PR2          - 17BCC SB%RD:(00005) Type=1.2 Nibs=5 Dist=14995
READP5 =  0323B JP%PR2          -
REAL   =  0AC02 AB%REG          - 1EC3E SB%TAB(00707) Type=1.2 Nibs=5 Dist=1403C
```

```
RECADR =  0F4B7 AB%EXP        - 0BBF3 AB%FCN(005BA) Type=1.1 Nibs=4 Dist=038C4
                              + 13EB1 SC%DAT(005FC) Type=1.1 Nibs=4 Dist=049FA
RECALL =  0F281 AB%EXP        - 199FB SC%SUB(00C89) Type=0.1 Nibs=5
                              + 19E3F SC%SUB(010CD) Type=0.1 Nibs=5
RED    =  0C12A AB%FCN        - 1DFEE JP%TAB(002EA) Type=1.2 Nibs=5 Dist=11EC4
REDNUM =  13C03 SC%DAT        -
REDPE5 =  0326B JP%PR2        - 039F6 JP%PR3(00620) Type=1.0 Nibs=3 Dist=0078B
REDREC =  13B6B SC%DAT        -
REDUCE =  15977 AB%CLC        - 1677A AB%ED:(00108) Type=1.1 Nibs=4 Dist=00E03
REDUCe =  0C852 JT%MTH        -
REL3D0 =  023FD SB%DSP        - 02CC0 JP%PR2(0009A) Type=1.0 Nibs=4 Dist=008C3
RELJMP =  05047 SG%LDC        - 05DF6 SB%EXD(004D4) Type=1.0 Nibs=4 Dist=00DAF
REM    =  08A48 SG%EXC        - 1EDB8 SB%TAB(00881) Type=1.2 Nibs=5 Dist=16370
REM12  =  0C7CF JT%MTH        - 0C132 AB%FCN(00AF9) Type=1.1 Nibs=3 Dist=0069D
REM15  =  0C7D3 JT%MTH        -
REMDC  =  05352 SG%LDC        - 08A3E SG%EXC(003F6) Type=1.2 Nibs=5 Dist=036EC
REMP   =  03548 JP%PR3        - 02E28 JP%PR2(00202) Type=1.0 Nibs=3 Dist=00720
                              + 08A43 SG%EXC(003FB) Type=1.2 Nibs=5 Dist=054FB
REMP05 =  03551 JP%PR3        - 058AC SG%LDC(0094E) Type=1.1 Nibs=4 Dist=0235B
REMP10 =  03554 JP%PR3        - 05354 SG%LDC(003F6) Type=1.1 Nibs=4 Dist=01E00
REN180 =  1A70C SC%REN        - 1A44B SB%EXC(00077) Type=1.0 Nibs=3 Dist=002C1
                              + 1AD2C SG%KEY(0028B) Type=1.0 Nibs=3 Dist=00620
                              + 1C69B SG%POK(001AD) Type=1.0 Nibs=4 Dist=01F8F
                              + 1CECA MN%CD:(00651) Type=1.0 Nibs=4 Dist=027BE
RENAME =  081BA JP%EXC        - 1DFF7 JP%TAB(002F3) Type=1.2 Nibs=5 Dist=15E3D
RENAMP =  03B1C JP%PR3        - 081B5 JP%EXC(003D9) Type=1.2 Nibs=5 Dist=04699
RENMDC =  05726 SG%LDC        - 081B0 JP%EXC(003D4) Type=1.2 Nibs=5 Dist=02A8A
RENMP  =  03C4B JP%PR3        - 1A581 SC%REN(00005) Type=1.2 Nibs=5 Dist=16936
RENSUB =  1A753 SC%REN        -
RENUM  =  1A586 SC%REN        - 1E000 JP%TAB(002FC) Type=1.2 Nibs=5 Dist=03A7A
REPCUR =  00787 SB%BIT        - 01B45 SB%DSP(002B5) Type=0.0 Nibs=5
REPROM =  18A1E SB%IO:        -
RES    =  0B91F AB%FCN        - 1EA19 SB%TAB(004E2) Type=1.2 Nibs=5 Dist=130FA
RESCAN =  04A4C AB%LEX        - 02C71 JP%PR2(0004B) Type=1.1 Nibs=4 Dist=01DDB
                              + 02FE4 JP%PR2(003BE) Type=1.1 Nibs=4 Dist=01A68
RESERV =  2F986 SB%RAM        -
RESET  =  0EDAB MN%UTL        - 1E009 JP%TAB(00305) Type=1.2 Nibs=5 Dist=0F25E
RESETC =  128F8 MN%TM:        - 0EDC2 MN%UTL(00154) Type=1.1 Nibs=4 Dist=03B36
RESETd =  05911 SG%LDC        - 0EDA1 MN%UTL(00133) Type=1.2 Nibs=5 Dist=09490
RESETp =  03C69 JP%PR3        - 0EDA6 MN%UTL(00138) Type=1.2 Nibs=5 Dist=0B13D
RESPTR =  03172 JP%PR2        - 02A38 JP%PR1(00498) Type=1.1 Nibs=3 Dist=0073A
                              + 02A75 JP%PR1(004D5) Type=1.0 Nibs=3 Dist=006FD
                              + 02A7B JP%PR1(004DB) Type=1.1 Nibs=3 Dist=006F7
                              + 02AC3 JP%PR1(00523) Type=1.1 Nibs=3 Dist=006AF
                              + 02B1D JP%PR1(0057D) Type=1.1 Nibs=3 Dist=00655
                              + 02B39 JP%PR1(00599) Type=1.1 Nibs=3 Dist=00639
                              + 02BAD JP%PR1(0060D) Type=1.0 Nibs=3 Dist=005C5
                              + 03412 JP%PR3(0003C) Type=1.0 Nibs=3 Dist=002A0
                              + 03523 JP%PR3(0014D) Type=1.1 Nibs=3 Dist=003B1
                              + 035D4 JP%PR3(001FE) Type=1.1 Nibs=3 Dist=00462
                              + 035F0 JP%PR3(0021A) Type=1.0 Nibs=3 Dist=0047E
                              + 037A2 JP%PR3(003CC) Type=1.0 Nibs=3 Dist=00630
                              + 040BA SB%EXP(000E5) Type=1.1 Nibs=4 Dist=00F48
                              + 04570 SB%EXP(0059B) Type=1.1 Nibs=4 Dist=013FE
                              + 18831 SB%IO:(00A75) Type=0.1 Nibs=5
                              + 188FF SB%IO:(00B43) Type=0.1 Nibs=5
RESREG =  2F7C2 SB%RAM        - 0B924 AB%FCN(002EB) Type=0.0 Nibs=5
                              + 0F624 AB%ASN(0004A) Type=0.0 Nibs=5
                              + 15DD4 AB%CLC(008B5) Type=0.0 Nibs=5
                              + 15E4C AB%CLC(0092D) Type=0.0 Nibs=5
```

```
                                + 18120 SB%IO:(00364) Type=0.0 Nibs=5
REST*  =  03035 JP%PR2          -
RESTDO =  1129D SC%FIL          - 13D69 SC%DAT(004B4) Type=1.1 Nibs=4 Dist=02ACC
                                + 17D9F SB%RD:(001D8) Type=1.1 Nibs=4 Dist=06B02
                                + 18F61 SC%SUB(001EF) Type=1.1 Nibs=4 Dist=07CC4
RESTDC =  0552E SG%LDC          - 079B0 JP%SYS(00A81) Type=1.2 Nibs=5 Dist=02482
RESTOR =  079BA JP%SYS          - 1ED70 SB%TAB(00839) Type=1.2 Nibs=5 Dist=173B6
RESTR# =  111AC SC%FIL          - 079C8 JP%SYS(00A99) Type=0.1 Nibs=5
RESTRP =  02A52 JP%PR1          - 079B5 JP%SYS(00A86) Type=1.2 Nibs=5 Dist=04F63
RETRNp =  03C62 JP%PR3          - 08FDB SG%EXC(00993) Type=1.2 Nibs=5 Dist=05379
RETURN =  08FE0 SG%EXC          - 1ED55 SB%TAB(0081E) Type=1.2 Nibs=5 Dist=15D75
REV$   =  1B38E AB%UTL          - 08B0D SG%EXC(004C5) Type=0.1 Nibs=5
                                + 0BD33 AB%FCN(006FA) Type=0.1 Nibs=5
                                + 180E3 SB%IO:(00327) Type=1.1 Nibs=4 Dist=032AB
                                + 18C10 SB%IO:(00E54) Type=1.1 Nibs=4 Dist=0277E
                                + 1C56D SG%POK(0007F) Type=1.1 Nibs=4 Dist=011DF
REVPOP =  0BD31 AB%FCN          - 09B9A SG%FXQ(00024) Type=1.1 Nibs=4 Dist=02197
                                + 0FDDA SC%TRC(00393) Type=1.1 Nibs=4 Dist=040A9
                                + 16345 AB%BLD(00019) Type=0.1 Nibs=5
                                + 186D3 SB%IO:(00917) Type=0.1 Nibs=5
                                + 18C68 MN%LCK(00010) Type=0.1 Nibs=5
                                + 18D16 MN%LCK(000BE) Type=0.1 Nibs=5
                                + 1A540 SB%EXC(0016C) Type=0.1 Nibs=5
                                + 1ADA8 SB%VAL(0003A) Type=0.1 Nibs=5
                                + 1B503 MB%IMG(000BD) Type=0.1 Nibs=5
                                + 1BC50 MB%USG(001AC) Type=0.1 Nibs=5
                                + 1BCB0 MB%USG(0020C) Type=0.1 Nibs=5
                                + 1C5A7 SG%POK(000B9) Type=0.1 Nibs=5
REWIND =  11365 SC%FIL          - 173A9 FH%TFM(008FB) Type=1.1 Nibs=4 Dist=06044
RFAD++ =  0A6FB SG%FXQ          - 08C37 SG%EXC(005EF) Type=1.1 Nibs=4 Dist=01AC4
RFAD+I =  0A702 SG%FXQ          - 013C1 TI%UTL(00290) Type=0.1 Nibs=5
                                + 14B6E SC%DAT(012B9) Type=0.1 Nibs=5
RFAD-- =  0A652 SG%FXQ          - 08D5A SG%EXC(00712) Type=1.0 Nibs=4 Dist=018F8
RFAD-I =  0A659 SG%FXQ          - 013C8 TI%UTL(00297) Type=0.1 Nibs=5
                                + 170F4 FH%TFM(00646) Type=0.1 Nibs=5
RFADJ+ =  0A6F8 SG%FXQ          - 0851B JP%EXC(0073F) Type=1.1 Nibs=4 Dist=021DD
                                + 10CD7 MN%CNF(00B45) Type=1.1 Nibs=4 Dist=065DF
RFNBFR =  2F57B SB%RAM          - 155A8 AB%CLC(00089) Type=0.0 Nibs=5
                                + 1586E AB%CLC(0034F) Type=0.0 Nibs=5
                                + 15979 AB%CLC(0045A) Type=0.0 Nibs=5
                                + 15A7B AB%CLC(0055C) Type=0.0 Nibs=5
                                + 15BD8 AB%CLC(006B9) Type=0.0 Nibs=5
                                + 15D0F AB%CLC(007F0) Type=0.0 Nibs=4
                                + 16209 AB%CLC(00CEA) Type=0.0 Nibs=4
                                + 168B8 AB%ED:(00246) Type=0.0 Nibs=5
                                + 168EC AB%ED:(0027A) Type=0.0 Nibs=4
RFPROT =  1C7AF SG%POK          -
RFUPD+ =  0A66E SG%FXQ          - 14B7C SC%DAT(012C7) Type=0.1 Nibs=5
                                + 14B86 SC%DAT(012D1) Type=0.1 Nibs=5
RJUST  =  12AE2 MN%TM:          - 0EB09 MN%BP:(00118) Type=1.1 Nibs=4 Dist=03FD9
                                + 0EB29 MN%BP:(00138) Type=1.1 Nibs=4 Dist=03FB9
RLINFO =  015E5 TI%UTL          - 08403 JP%EXC(00627) Type=1.0 Nibs=4 Dist=06E1E
                                + 16CCF FH%TFM(00221) Type=0.1 Nibs=5
RMD    =  0C104 AB%FCN          - 1E977 SB%TAB(00440) Type=1.2 Nibs=5 Dist=12873
RMD12  =  0C77A JT%MTH          - 0C10C AB%FCN(00AD3) Type=1.1 Nibs=3 Dist=0066E
RMD15  =  0C77E JT%MTH          -
RMEM   =  12325 JP%POL          -
RMEM10 =  013EF TI%UTL          - 0853F JP%EXC(00763) Type=1.0 Nibs=4 Dist=07150
RMEMCH =  013CD TI%UTL          - 01387 TI%UTL(00256) Type=1.1 Nibs=3 Dist=00046
                                + 06D2B SG%SYS(0098B) Type=1.1 Nibs=4 Dist=0595E
```

```
RND    =  0E6BF PM%STA       - 1EB42 SB%TAB(0060B) Type=1.2 Nibs=5 Dist=10483
RND-12 =  1B01F AB%UTL       - 0F9AE AB%ASN(003D4) Type=0.1 Nibs=5
                             + 1BDB0 MB%USG(0030C) Type=1.1 Nibs=4 Dist=00D91
RND12+ =  0C9D5 JT%MTH       -
RNDAHX =  136CB PM%FLG       -
RNDEXP =  080A0 JP%EXC       -
RNDNRM =  0CAB1 JT%MTH       -
RNDXCP =  0C9DE JT%MTH       -
RNM010 =  0A26F SG%FXQ       - 08265 JP%EXC(00489) Type=1.0 Nibs=4 Dist=0200A
RNMERR =  0A31A SG%FXQ       - 11609 SC%FIL(005B6) Type=1.0 Nibs=4 Dist=072EF
RNSEED =  2F6FE SB%RAM       - 002DE SB%DVR(002DE) Type=0.0 Nibs=4
                             + 0E6AA PM%STA(0087B) Type=0.0 Nibs=5
                             + 0E6C8 PM%STA(00899) Type=0.0 Nibs=5
RNUMDC =  05839 SG%LDC       - 1A57C SC%REN(00000) Type=1.2 Nibs=5 Dist=14D43
ROMCHK =  10FDE MN%CNF       - 0A449 SG%FXQ(008D3) Type=1.0 Nibs=4 Dist=06B95
                             + 1A269 SC%SUB(014F7) Type=0.1 Nibs=5
ROMCID =  00BFE TI%EQU       - 10FE2 MN%CNF(00E50) Type=0.0 Nibs=2
ROMF-  =  0A01F SG%FXQ       - 011A2 TI%UTL(00071) Type=0.1 Nibs=5
                             + 0BFA6 AB%FCN(0096D) Type=1.1 Nibs=4 Dist=01F87
                             + 14AA9 SC%DAT(011F4) Type=0.1 Nibs=5
ROMF-1 =  0A01C SG%FXQ       - 06602 SG%SYS(00262) Type=1.1 Nibs=4 Dist=03A1A
                             + 06717 SG%SYS(00377) Type=1.1 Nibs=4 Dist=03905
                             + 0856B JP%EXC(0078F) Type=1.1 Nibs=4 Dist=01AB1
ROMFND =  1102F MN%CNF       - 0A44F SG%FXQ(008D9) Type=1.0 Nibs=4 Dist=06BE0
                             + 1A281 SC%SUB(0150F) Type=0.1 Nibs=5
ROMTST =  02484 MB%ROM       - 010A2 SB%KEY(003AB) Type=1.1 Nibs=4 Dist=013E2
ROWDVR =  2E350 SB%RAM       - 00667 SB%DVR(00667) Type=0.0 Nibs=5
RPLLIN =  013F7 TI%UTL       - 0FFB8 JP%MEM(00059) Type=0.1 Nibs=5
                             + 17561 FH%TFM(00AB3) Type=0.1 Nibs=5
RPLSBH =  1799B FH%TFM       -
RPTKY  =  152BA MN%ED:       - 0016C SB%DVR(0016C) Type=0.1 Nibs=5
                             + 021CD SB%DSP(0093D) Type=0.1 Nibs=5
                             + 02212 SB%DSP(00982) Type=0.1 Nibs=5
                             + 064C8 SG%SYS(00128) Type=0.1 Nibs=5
RPTNaN =  0E1D0 PM%STA       -
RRADD  =  173BB FH%TFM       -
RS-IDS =  0F156 MN%UTL       - 10EC0 MN%CNF(00D2E) Type=1.1 Nibs=4 Dist=01D6A
RS-R03 =  02C13 JP%PR1       - 03083 JP%PR2(0045D) Type=1.1 Nibs=3 Dist=00470
RST2<R =  014A6 TI%UTL       - 0474F SB%EXP(0077A) Type=1.1 Nibs=4 Dist=032A9
                             + 09616 TI%ERD(00283) Type=0.1 Nibs=5
                             + 1CF3A MN%CD:(006C1) Type=0.1 Nibs=5
RST<R3 =  109DA MN%CNF       - 10F0D MN%CNF(00D7B) Type=1.1 Nibs=3 Dist=00533
RSTD0  =  06832 SG%SYS       - 0E3D5 PM%STA(005A6) Type=1.1 Nibs=4 Dist=07BA3
RSTD1  =  1C596 SG%POK       - 08C46 SG%EXC(005FE) Type=0.1 Nibs=5
RSTK<R =  014A8 TI%UTL       - 01C4B SB%DSP(003BB) Type=1.1 Nibs=3 Dist=007A3
                             + 1246D JP%POL(00152) Type=0.1 Nibs=5
RSTKBF =  2F820 SB%RAM       - 014BF TI%UTL(0038E) Type=0.0 Nibs=2 Offset=     75
                             + 014F4 TI%UTL(003C3) Type=0.0 Nibs=2
RSTKBp =  2F81F SB%RAM       - 014CD TI%UTL(0039C) Type=0.0 Nibs=2
                             + 01508 TI%UTL(003D7) Type=0.0 Nibs=5
RSTOPT =  1A16E SC%SUB       - 07DB4 JP%SYS(00E85) Type=0.1 Nibs=5
                             + 07DBE JP%SYS(00E8F) Type=0.1 Nibs=5
RSTPR1 =  029D0 JP%PR1       -
RSTST  =  0F5C5 AB%EXP       - 08792 SG%EXC(0014A) Type=1.1 Nibs=4 Dist=06E33
                             + 09F29 SG%FXQ(003B3) Type=1.0 Nibs=4 Dist=0569C
                             + 0AC4C AB%REG(00054) Type=1.0 Nibs=4 Dist=04979
                             + 16322 AB%CLC(00E03) Type=1.0 Nibs=4 Dist=06D5D
RTD    =  0C155 AB%FCN       -
RTNCC  =  03FB2 JP%PR3       - 07691 JP%SYS(00762) Type=1.2 Nibs=5 Dist=036DF
                             + 076CD JP%SYS(0079E) Type=1.2 Nibs=5 Dist=0371B
```

```
                                + 07FDD JP%EXC(00201) Type=1.2 Nibs=5 Dist=0402B
                                + 08FCA SG%EXC(00982) Type=1.2 Nibs=5 Dist=05018
                                + 0DE49 PM%STA(0001A) Type=1.2 Nibs=5 Dist=09E97
                                + 0EFF2 MN%UTL(00384) Type=1.2 Nibs=5 Dist=0B040
                                + 12DD1 MN%TM:(00887) Type=1.2 Nibs=5 Dist=0EE1F
                                + 1385E PM%FLG(00415) Type=1.2 Nibs=5 Dist=0F8AC
                                + 13870 PM%FLG(00427) Type=1.2 Nibs=5 Dist=0F8BE
                                + 1D710 MN%CD:(00E97) Type=1.2 Nibs=5 Dist=1975E
RTNHER =  1AF32 MN%GSB          - 1AF1E MN%GSB(0001D) Type=0.0 Nibs=5
RTNSET =  02603 JP%PR1          - 04F62 SG%LDC(00004) Type=1.1 Nibs=4 Dist=0295F
RTNSXM =  1244D JP%POL          - 10E3A MN%CNF(00CA8) Type=0.0 Nibs=5
                                + 1DD67 JP%TAB(00063) Type=1.2 Nibs=5 Dist=0B91A
                                + 1E59A SB%TAB(00063) Type=1.2 Nibs=5 Dist=0C14D
RUN    =  06F6E JP%SYS          - 1EE90 SB%TAB(00959) Type=1.2 Nibs=5 Dist=17F22
RUNDC  =  05508 SG%LDC          - 06F2F JP%SYS(00000) Type=1.2 Nibs=5 Dist=01A27
                                + 06F64 JP%SYS(00035) Type=1.2 Nibs=5 Dist=01A5C
RUNER1 =  0717B JP%SYS          -
RUNK   =  06F47 JP%SYS          - 004B1 SB%DVR(004B1) Type=1.0 Nibs=4 Dist=06A96
RUNP   =  03DB1 JP%PR3          - 06F69 JP%SYS(0003A) Type=1.2 Nibs=5 Dist=031B8
RUNRT1 =  074E7 JP%SYS          - 19655 SC%SUB(008E3) Type=0.1 Nibs=5
RUNRTN =  074EA JP%SYS          - 08A67 SG%EXC(0041F) Type=1.0 Nibs=4 Dist=0157D
RUSUS? =  07B71 JP%SYS          - 18FCA SC%SUB(00258) Type=0.1 Nibs=5
Range  =  045D7 SB%EXP          - 003D4 SB%DVR(003D4) Type=1.1 Nibs=4 Dist=04203
                                + 03CDF JP%PR3(00909) Type=1.1 Nibs=4 Dist=008F8
                                + 03F95 JP%PR3(00BBF) Type=1.1 Nibs=3 Dist=00642
                                + 04973 AB%LEX(0008D) Type=1.1 Nibs=3 Dist=0039C
                                + 04988 AB%LEX(000A2) Type=1.1 Nibs=3 Dist=003B1
                                + 04A13 AB%LEX(0012D) Type=1.1 Nibs=3 Dist=0043C
                                + 04C9D AB%LEX(003B7) Type=1.1 Nibs=3 Dist=006C6
                                + 053B5 SG%LDC(00457) Type=1.1 Nibs=4 Dist=00DDE
                                + 05FB9 SB%EXD(00697) Type=1.0 Nibs=4 Dist=019E2
                                + 08169 JP%EXC(0038D) Type=1.1 Nibs=4 Dist=03B92
                                + 099FE TI%ERD(0066B) Type=1.1 Nibs=4 Dist=05427
Rangej =  15F42 AB%CLC          - 164D9 AB%BLD(001AD) Type=1.1 Nibs=3 Dist=00597
                                + 164E6 AB%BLD(001BA) Type=1.1 Nibs=3 Dist=005A4
ResetC =  00008 SB%DSP          -
RmTsMs =  02584 MB%ROM          - 02486 MB%ROM(00002) Type=0.0 Nibs=5
Rstk<R =  1246B JP%POL          - 109DE MN%CNF(0084C) Type=1.0 Nibs=4 Dist=01A8D
                                + 1728E FH%TFM(007E0) Type=1.0 Nibs=4 Dist=04E23

S-CRLF =  022A2 SB%DSP          - 0A4D6 SG%FXQ(00960) Type=0.1 Nibs=5
S-R0-0 =  2F871 SB%RAM          - 02DE2 JP%PR2(001BC) Type=0.0 Nibs=4
                                + 02EBA JP%PR2(00294) Type=0.0 Nibs=5
                                + 064DA SG%SYS(0013A) Type=0.0 Nibs=5
                                + 07A87 JP%SYS(00B58) Type=0.0 Nibs=5
                                + 07B3A JP%SYS(00C0B) Type=0.0 Nibs=5
                                + 0887E SG%EXC(00236) Type=0.0 Nibs=5
                                + 08AF1 SG%EXC(004A9) Type=0.0 Nibs=5
                                + 08B49 SG%EXC(00501) Type=0.0 Nibs=5
                                + 0A0CC SG%FXQ(00556) Type=0.0 Nibs=5
                                + 0A0F6 SG%FXQ(00580) Type=0.0 Nibs=5
                                + 0A241 SG%FXQ(006CB) Type=0.0 Nibs=4
                                + 0E051 PM%STA(00222) Type=0.0 Nibs=5
                                + 0E07E PM%STA(0024F) Type=0.0 Nibs=5
                                + 0F601 AB%ASN(00027) Type=0.0 Nibs=5
                                + 0F6BB AB%ASN(000E1) Type=0.0 Nibs=5
                                + 0F7C8 AB%ASN(001EE) Type=0.0 Nibs=5
                                + 0F926 AB%ASN(0034C) Type=0.0 Nibs=5
                                + 0FABB SC%TRC(00074) Type=0.0 Nibs=5
                                + 0FC81 SC%TRC(0023A) Type=0.0 Nibs=5
```

```
                                + 0FD7F SC%TRC(00338) Type=0.0 Nibs=5
                                + 1153A SC%FIL(004E7) Type=0.0 Nibs=5
                                + 1166C SC%FIL(00619) Type=0.0 Nibs=5
                                + 147B9 SC%DAT(00F04) Type=0.0 Nibs=5
                                + 14806 SC%DAT(00F51) Type=0.0 Nibs=5
                                + 17A14 FH%TFM(00F66) Type=0.0 Nibs=5
                                + 1900D SC%SUB(0029B) Type=0.0 Nibs=5
                                + 1920F SC%SUB(0049D) Type=0.0 Nibs=5
                                + 19F22 SC%SUB(011B0) Type=0.0 Nibs=5
                                + 1C633 SG%POK(00145) Type=0.0 Nibs=5
                                + 1C65A SG%POK(0016C) Type=0.0 Nibs=5
S-R0-1 =  2F876 SB%RAM          - 0647B SG%SYS(000DB) Type=0.0 Nibs=5
                                + 064A2 SG%SYS(00102) Type=0.0 Nibs=5
                                + 06539 SG%SYS(00199) Type=0.0 Nibs=5
                                + 0711B JP%SYS(001EC) Type=0.0 Nibs=5
                                + 0721E JP%SYS(002EF) Type=0.0 Nibs=5
                                + 0A171 SG%FXQ(005FB) Type=0.0 Nibs=5
                                + 0A1EE SG%FXQ(00678) Type=0.0 Nibs=5
                                + 0ACC8 AB%REG(000D0) Type=0.0 Nibs=5
                                + 0DF69 PM%STA(0013A) Type=0.0 Nibs=5
                                + 0E061 PM%STA(00232) Type=0.0 Nibs=5
                                + 0E0AF PM%STA(00280) Type=0.0 Nibs=5
                                + 0FD26 SC%TRC(002DF) Type=0.0 Nibs=5
                                + 170E8 FH%TFM(0063A) Type=0.0 Nibs=5
S-R0-2 =  2F87B SB%RAM          - 02605 JP%PR1(00065) Type=0.0 Nibs=5
                                + 02C04 JP%PR1(00664) Type=0.0 Nibs=5
                                + 0648B SG%SYS(000EB) Type=0.0 Nibs=5
                                + 06591 SG%SYS(001F1) Type=0.0 Nibs=5
S-R0-3 =  2F880 SB%RAM          - 02C18 JP%PR1(00678) Type=0.0 Nibs=5
                                + 063A2 SG%SYS(00002) Type=0.0 Nibs=5
                                + 06612 SG%SYS(00272) Type=0.0 Nibs=5
                                + 0AC8D AB%REG(00095) Type=0.0 Nibs=5
                                + 0E0C3 PM%STA(00294) Type=0.0 Nibs=5
                                + 13E7E SC%DAT(005C9) Type=0.0 Nibs=5
                                + 14865 SC%DAT(00FB0) Type=0.0 Nibs=5
S-R1-0 =  2F881 SB%RAM          - 025B1 JP%PR1(00011) Type=0.0 Nibs=5
                                + 09D29 SG%FXQ(001B3) Type=0.0 Nibs=5
                                + 09D49 SG%FXQ(001D3) Type=0.0 Nibs=5
                                + 0F7DD AB%ASN(00203) Type=0.0 Nibs=4
                                + 0F8F1 AB%ASN(00317) Type=0.0 Nibs=5
                                + 0FC49 SC%TRC(00202) Type=0.0 Nibs=5
                                + 11764 SC%FIL(00711) Type=0.0 Nibs=5 Offset=      5
                                + 117CC SC%FIL(00779) Type=0.0 Nibs=5 Offset=      4
                                + 14A84 SC%DAT(011CF) Type=0.0 Nibs=5
                                + 19D15 SC%SUB(00FA3) Type=0.0 Nibs=5
S-R1-1 =  2F886 SB%RAM          - 06ADA SG%SYS(0073A) Type=0.0 Nibs=5
                                + 06B48 SG%SYS(007A8) Type=0.0 Nibs=5
                                + 0F7BB AB%ASN(001E1) Type=0.0 Nibs=5
                                + 14AC1 SC%DAT(0120C) Type=0.0 Nibs=5
                                + 14AF9 SC%DAT(01244) Type=0.0 Nibs=5
                                + 14B67 SC%DAT(012B2) Type=0.0 Nibs=5
                                + 1729E FH%TFM(007F0) Type=0.0 Nibs=5
                                + 1ACFC SG%KEY(0025B) Type=0.0 Nibs=5
S-R1-2 =  2F88B SB%RAM          - 025D7 JP%PR1(00037) Type=0.0 Nibs=5
                                + 08911 SG%EXC(002C9) Type=0.0 Nibs=5
                                + 0ADF9 AB%REG(00201) Type=0.0 Nibs=5
                                + 0AE46 AB%REG(0024E) Type=0.0 Nibs=5
                                + 0FA3A AB%ASN(00460) Type=0.0 Nibs=5
                                + 0FE0B SC%TRC(003C4) Type=0.0 Nibs=5
                                + 11573 SC%FIL(00520) Type=0.0 Nibs=5
```

```
                                    + 1558F AB%CLC(00070) Type=0.0 Nibs=5
                                    + 15728 AB%CLC(00209) Type=0.0 Nibs=5 Offset=      4
                                    + 15810 AB%CLC(002F1) Type=0.0 Nibs=5
                                    + 15C76 AB%CLC(00757) Type=0.0 Nibs=5 Offset=      2
                                    + 167AC AB%ED:(0013A) Type=0.0 Nibs=5 Offset=      2
                                    + 167F6 AB%ED:(00184) Type=0.0 Nibs=5 Offset=      4
                                    + 16922 AB%ED:(002B0) Type=0.0 Nibs=4
                                    + 16959 AB%ED:(002E7) Type=0.0 Nibs=5 Offset=      4
                                    + 199D7 SC%SUB(00C65) Type=0.0 Nibs=5
                                    + 19DF3 SC%SUB(01081) Type=0.0 Nibs=5
S-R1-3 =  2F890 SB%RAM              - 029D2 JP%PR1(00432) Type=0.0 Nibs=5
                                    + 0300D JP%PR2(003E7) Type=0.0 Nibs=5
                                    + 03037 JP%PR2(00411) Type=0.0 Nibs=5
                                    + 0AC1B AB%REG(00023) Type=0.0 Nibs=5
                                    + 0F869 AB%ASN(0028F) Type=0.0 Nibs=5
SALLOC =  0153B TI%UTL              - 12367 JP%POL(0004C) Type=0.1 Nibs=5
SAVDO  =  1C587 SG%POK              - 0666A SG%SYS(002CA) Type=0.1 Nibs=5
                                    + 0E54C PM%STA(0071D) Type=0.1 Nibs=5
SAVDO- =  1149D SC%FIL              -
SAVD1  =  1C578 SG%POK              - 0BF9F AB%FCN(00966) Type=0.1 Nibs=5
SAVEDO =  1149A SC%FIL              - 09B7A SG%FXQ(00004) Type=1.1 Nibs=4 Dist=07920
                                    + 141F3 SC%DAT(0093E) Type=1.0 Nibs=4 Dist=02D59
                                    + 18DF7 SC%SUB(00085) Type=1.1 Nibs=4 Dist=0795D
SAVEL# =  10178 JP%MEM              - 07392 JP%SYS(00463) Type=0.1 Nibs=5
                                    + 09136 SG%EXC(00AEE) Type=1.1 Nibs=4 Dist=07042
SAVEL+ =  10175 JP%MEM              -
SAVELO =  1017E JP%MEM              - 09704 TI%ERD(00371) Type=1.0 Nibs=4 Dist=06A7A
                                    + 0A515 SG%FXQ(0099F) Type=1.1 Nibs=4 Dist=05C69
SAVESB =  0D66E SM%MTH              - 0E19F PM%STA(00370) Type=1.1 Nibs=4 Dist=00B31
SAVEXM =  0D663 SM%MTH              -
SAVGSB =  0D64E SM%MTH              -
SAVSTK =  2F59E SB%RAM              - 0154F TI%UTL(0041E) Type=0.0 Nibs=5
                                    + 08470 JP%EXC(00694) Type=0.0 Nibs=5
                                    + 124FA JP%POL(001DF) Type=0.0 Nibs=5
                                    + 17DE8 SB%IO:(0002C) Type=0.0 Nibs=5
SB15S  =  0E19A PM%STA              -
SBEXC  =  1A3D4 SB%EXC              -
SBFCN  =  0A92B SB%FCN              -
SCAN   =  04C40 AB%LEX              -
SCNRT  =  022B9 SB%DSP              -
SCOPCK =  0915B SG%EXC              - 076C4 JP%SYS(00795) Type=1.0 Nibs=4 Dist=01A97
                                    + 19603 SC%SUB(00891) Type=0.1 Nibs=5
SCOPEN =  07BEA JP%SYS              - 1A212 SC%SUB(014A0) Type=0.1 Nibs=5
SCREX0 =  2F941 SB%RAM              - 13160 MN%TM:(00C16) Type=0.0 Nibs=5
                                    + 1319B MN%TM:(00C51) Type=0.0 Nibs=5
SCREX1 =  2F951 SB%RAM              -
SCREX2 =  2F961 SB%RAM              -
SCREX3 =  2F971 SB%RAM              -
SCRLLR =  0212E SB%DSP              - 00173 SB%DVR(00173) Type=0.1 Nibs=5
                                    + 064D2 SG%SYS(00132) Type=1.1 Nibs=4 Dist=043A4
                                    + 16360 AB%BLD(00034) Type=0.1 Nibs=5
                                    + 1D05E MN%CD:(007E5) Type=0.1 Nibs=5
SCROLT =  2F946 SB%RAM              - 00223 SB%DVR(00223) Type=0.0 Nibs=4
                                    + 01CF2 SB%DSP(00462) Type=0.0 Nibs=4
                                    + 01D7D SB%DSP(004ED) Type=0.0 Nibs=4
                                    + 0EED9 MN%UTL(0026B) Type=0.0 Nibs=4
SCRPTR =  2F966 SB%RAM              - 0E9CC PM%STA(00B9D) Type=0.0 Nibs=5
SCRST0 =  2F901 SB%RAM              - 0CC72 JT%MTH(0094B) Type=0.0 Nibs=5
                                    + 0CCBC JT%MTH(00995) Type=0.0 Nibs=5 Offset=     64
                                    + 0E9D5 PM%STA(00BA6) Type=0.0 Nibs=5
```

```
SCRTCH =  2F901 SB%RAM      - 013B0 TI%UTL(0027F) Type=0.0 Nibs=5
                            + 11CDC SC%FIL(00C89) Type=0.0 Nibs=5 Offset=     56
                            + 11E3D SC%FIL(00DEA) Type=0.0 Nibs=5 Offset=     33
                            + 11E92 SC%FIL(00E3F) Type=0.0 Nibs=4 Offset=     56
                            + 12AAE MN%TM:(00564) Type=0.0 Nibs=5
                            + 12AC8 MN%TM:(0057E) Type=0.0 Nibs=5
                            + 1AF64 MN%GSB(00063) Type=0.0 Nibs=5 Offset=     64
                            + 1AF9A MN%GSB(00099) Type=0.0 Nibs=4
                            + 1AFC1 MN%GSB(000C0) Type=0.0 Nibs=5 Offset=     64
                            + 1AFF2 MN%GSB(000F1) Type=0.0 Nibs=4 Offset=     80
                            + 1CF13 MN%CD:(0069A) Type=0.0 Nibs=5 Offset=     32
                            + 1D021 MN%CD:(007A8) Type=0.0 Nibs=5 Offset=     32
SDEV   =  0E2EF PM%STA      - 1E01B JP%TAB(00317) Type=1.2 Nibs=5 Dist=0FD2C
                            + 1EB30 SB%TAB(005F9) Type=1.2 Nibs=5 Dist=10841
SE1-10 =  04468 SB%EXP      -
SEC2TK =  13203 MN%TM:      - 07E93 JP%EXC(000B7) Type=0.1 Nibs=5
                            + 0ED62 MN%UTL(000F4) Type=1.1 Nibs=4 Dist=044A1
SECHMS =  13252 MN%TM:      -
SECRDC =  05839 SG%LDC      - 0A386 SG%FXQ(00810) Type=1.2 Nibs=5 Dist=04B4D
                            + 0A397 SG%FXQ(00821) Type=1.2 Nibs=5 Dist=04B5E
SECURE =  0A3A1 SG%FXQ      - 1E02D JP%TAB(00329) Type=1.2 Nibs=5 Dist=13C8C
SECURP =  03B8F JP%PR3      - 0A38B SG%FXQ(00815) Type=1.2 Nibs=5 Dist=067FC
                            + 0A39C SG%FXQ(00826) Type=1.2 Nibs=5 Dist=0680D
SEND20 =  17DFA SB%IO:      -
SENDEL =  17DC1 SB%IO:      - 06B3E SG%SYS(0079E) Type=0.1 Nibs=5
                            + 1C13F MB%USG(0069B) Type=1.1 Nibs=4 Dist=0437E
                            + 1C1BB MB%USG(00717) Type=1.1 Nibs=4 Dist=043FA
SENDIT =  17DE3 SB%IO:      -
SENDWD =  17E15 SB%IO:      - 09780 TI%ERD(003ED) Type=0.1 Nibs=5
                            + 1BC24 MB%USG(00180) Type=1.0 Nibs=4 Dist=03E0F
SET-ST =  1B9CF MB%IMG      - 1BCF9 MB%USG(00255) Type=1.1 Nibs=3 Dist=0032A
SETALM =  1290D MN%TM:      -
SETALR =  12917 MN%TM:      - 0ED70 MN%UTL(00102) Type=1.1 Nibs=4 Dist=03BA7
SETDAT =  12D9C MN%TM:      - 1E03F JP%TAB(0033B) Type=1.2 Nibs=5 Dist=0B2A3
SETDGT =  0F045 MN%UTL      -
SETFMT =  0F01F MN%UTL      -
SETIME =  12825 MN%TM:      -
SETSB  =  0D641 SM%MTH      - 0CEEC JT%MTH(00BC5) Type=1.1 Nibs=3 Dist=00755
                            + 0D0DA JT%MTH(00DB3) Type=1.0 Nibs=3 Dist=00567
SETSU1 =  0736D JP%SYS      - 0912A SG%EXC(00AE2) Type=1.1 Nibs=4 Dist=01DBD
SETSUS =  07360 JP%SYS      -
SETTIM =  12C9A MN%TM:      - 1E048 JP%TAB(00344) Type=1.2 Nibs=5 Dist=0B3AE
SETTMO =  13158 MN%TM:      - 0213D SB%DSP(008AD) Type=0.1 Nibs=5
                            + 14E43 MN%ED:(001C3) Type=1.1 Nibs=4 Dist=01CEB
SETTRC =  0FA38 AB%ASN      - 187A0 SB%IO:(009E4) Type=0.1 Nibs=5
                            + 18C29 SB%IO:(00E6D) Type=0.1 Nibs=5
SETWRT =  13AF8 SC%DAT      - 17588 FH%TFM(00ADA) Type=1.1 Nibs=4 Dist=03A90
SFGPGM =  0734F JP%SYS      - 198EA SC%SUB(00B78) Type=0.1 Nibs=5
SFLAG  =  13453 PM%FLG      - 1EE75 SB%TAB(0093E) Type=1.2 Nibs=5 Dist=0BA22
SFLAG* =  135F3 PM%FLG      - 07E22 JP%EXC(00046) Type=0.1 Nibs=5
SFLAG? =  1364C PM%FLG      - 00196 SB%DVR(00196) Type=0.1 Nibs=5
                            + 0EB56 MN%BP:(00165) Type=1.1 Nibs=4 Dist=04AF6
                            + 0EBFA MN%BP:(00209) Type=1.1 Nibs=4 Dist=04A52
                            + 13177 MN%TM:(00C2D) Type=1.1 Nibs=3 Dist=004D5
                            + 15108 MN%ED:(0048B) Type=1.0 Nibs=4 Dist=01ABF
                            + 18CA2 MN%LCK(0004A) Type=1.1 Nibs=4 Dist=05656
                            + 19A3E SC%SUB(00CCC) Type=1.1 Nibs=4 Dist=063F2
SFLAGC =  13601 PM%FLG      - 004CE SB%DVR(004CE) Type=0.1 Nibs=5
                            + 0EA95 MN%BP:(000A4) Type=1.1 Nibs=4 Dist=04B6C
                            + 0EF57 MN%UTL(002E9) Type=1.1 Nibs=4 Dist=046AA
```

```
                                + 12909 MNXTM:(003BF) Type=1.0 Nibs=4 Dist=00CF8
                                + 15111 MNXED:(00491) Type=1.0 Nibs=4 Dist=01B10
                                + 16876 ABXED:(00204) Type=1.1 Nibs=4 Dist=03275
                                + 18CBF MNXLCK(00067) Type=1.1 Nibs=4 Dist=056BE
SFLAGS =  135FA PMXFLG          - 004D5 SBXDVR(004D5) Type=0.1 Nibs=5
                                + 0EA8B MNXBP:(0009A) Type=1.1 Nibs=4 Dist=04B6F
                                + 0EF4D MNXUTL(002DF) Type=1.1 Nibs=4 Dist=046AD
                                + 1529B MNXED:(0061B) Type=1.0 Nibs=4 Dist=01CA1
                                + 1556F ABXCLC(00050) Type=1.1 Nibs=4 Dist=01F75
                                + 161DE ABXCLC(00CBF) Type=1.1 Nibs=4 Dist=02BE4
                                + 16248 ABXCLC(00D29) Type=1.1 Nibs=4 Dist=02C4E
                                + 18E08 SCXSUB(00096) Type=1.1 Nibs=4 Dist=0580E
SFLAGT =  13608 PMXFLG          - 01641 SBXCMD(0001A) Type=0.1 Nibs=5
                                + 07E15 JPXEXC(00039) Type=0.1 Nibs=5
                                + 15249 MNXED:(005C9) Type=1.0 Nibs=4 Dist=01C41
SFLGDC =  054AF SGXLDC          - 13449 PMXFLG(00000) Type=1.2 Nibs=5 Dist=0DF9A
                                + 13488 PMXFLG(0003F) Type=1.2 Nibs=5 Dist=0DFD9
SFLGP  =  03C7B JPXPR3          - 1344E PMXFLG(00005) Type=1.2 Nibs=5 Dist=0F7D3
                                + 1348D PMXFLG(00044) Type=1.2 Nibs=5 Dist=0F812
SFlag? =  07B7A JPXSYS          - 093DB TIXERD(00048) Type=1.1 Nibs=4 Dist=01861
SFlagC =  1510F MNXED:          - 19154 SCXSUB(003E2) Type=1.1 Nibs=4 Dist=04045
SGN    =  0B8D8 ABXFCN          - 1EB4B SBXTAB(00614) Type=1.2 Nibs=5 Dist=13273
SHF10  =  0C486 JTXMTH          - 0DDB6 SMXMTH(00981) Type=1.1 Nibs=4 Dist=01930
SHFLAC =  0DB46 SMXMTH          - 0CFF0 JTXMTH(00CC9) Type=1.1 Nibs=4 Dist=00B56
SHFRAC =  0DB51 SMXMTH          -
SHFRBD =  0DB5F SMXMTH          -
SHFTKN =  04BE6 ABXLEX          -
SHORT  =  0AC02 ABXREG          - 1ECC5 SBXTAB(0078E) Type=1.2 Nibs=5 Dist=140C3
SHOW   =  0A466 SGXFXQ          - 1E051 JPXTAB(0034D) Type=1.2 Nibs=5 Dist=13BEB
SHOWp  =  03389 JPXPR2          - 0A461 SGXFXQ(008EB) Type=1.2 Nibs=5 Dist=070D8
SHRT   =  0F96C ABXASN          -
SHUTDN =  005E2 SBXDVR          -
SIGCHK =  0BD98 ABXFCN          - 0F99F ABXASN(003C5) Type=1.1 Nibs=4 Dist=03C07
SIGNAN =  0C916 JTXMTH          - 0B7B5 ABXFCN(0017C) Type=1.1 Nibs=4 Dist=01161
SIGTST =  0E636 PMXSTA          - 0BD64 ABXFCN(0072B) Type=1.1 Nibs=4 Dist=028D2
                                + 0BD77 ABXFCN(0073E) Type=1.1 Nibs=4 Dist=028BF
                                + 0BD9A ABXFCN(00761) Type=1.1 Nibs=4 Dist=0289C
SIN    =  0BE53 ABXFCN          - 1EAE8 SBXTAB(005B1) Type=1.2 Nibs=5 Dist=12C95
SIN12  =  0D716 SMXMTH          - 0BE5C ABXFCN(00823) Type=1.1 Nibs=4 Dist=018BA
SIN15  =  0D71A SMXMTH          -
SINP30 =  1886C SBXIO:          - 18858 SBXIO:(00A9C) Type=0.0 Nibs=5
SKIPDC =  057F6 SGXLDC          -
SKPBYT =  139FA SCXDAT          - 11352 SCXFIL(002FF) Type=1.1 Nibs=4 Dist=026A8
SLEEP  =  006C2 SBXDVR          - 01D9B SBXDSP(0050B) Type=1.1 Nibs=4 Dist=016D9
                                + 01E29 SBXDSP(00599) Type=1.1 Nibs=4 Dist=01767
                                + 02148 SBXDSP(008B8) Type=1.1 Nibs=4 Dist=01A86
                                + 14E49 MNXED:(001C9) Type=0.1 Nibs=5
SMALL  =  0B736 ABXFCN          -
SNAPBF =  2F7F0 SBXRAM          - 01573 TIXUTL(00442) Type=0.0 Nibs=5 Offset=     42
                                + 015AE TIXUTL(0047D) Type=0.0 Nibs=5
                                + 12377 JPXPOL(0005C) Type=0.0 Nibs=5
                                + 124D8 JPXPOL(001BD) Type=0.0 Nibs=5 Offset=     42
SNAPLC =  015B3 TIXUTL          - 0CC59 JTXMTH(00932) Type=0.1 Nibs=5
SNAPR* =  01578 TIXUTL          - 1245B JPXPOL(00140) Type=0.1 Nibs=5
SNAPRS =  01571 TIXUTL          - 12451 JPXPOL(00136) Type=0.1 Nibs=5
SNAPSV =  015A7 TIXUTL          - 115D2 SCXFIL(0057F) Type=0.1 Nibs=5
                                + 11AFB SCXFIL(00AA8) Type=0.1 Nibs=5
                                + 1233B JPXPOL(00020) Type=0.1 Nibs=5
                                + 124AE JPXPOL(00193) Type=0.1 Nibs=5
                                + 13ABA SCXDAT(00205) Type=0.1 Nibs=5
```

```
SNDWD+ =  17E1F SB%IO:        - 06B37 SG%SYS(00797) Type=0.1 Nibs=5
SNcrlf =  08C72 SG%EXC        - 07B49 JP%SYS(00C1A) Type=1.1 Nibs=4 Dist=01129
                              + 1A039 SC%SUB(012C7) Type=0.1 Nibs=5
SPACE  =  0AD9D AB%REG        -
SPLITA =  0C6BF JT%MTH        - 0C15D AB%FCN(00B24) Type=1.1 Nibs=3 Dist=00562
                              + 0C189 AB%FCN(00B50) Type=1.1 Nibs=3 Dist=00536
                              + 0D708 SM%MTH(002D3) Type=1.0 Nibs=4 Dist=01049
                              + 0E62F PM%STA(00800) Type=1.1 Nibs=4 Dist=01F70
                              + 0F970 AB%ASN(00396) Type=1.1 Nibs=4 Dist=032B1
SPLITC =  0C940 JT%MTH        -
SPLTAC =  0C934 JT%MTH        - 0B6A1 AB%FCN(00068) Type=1.1 Nibs=4 Dist=01293
                              + 0D70E SM%MTH(002D9) Type=1.0 Nibs=4 Dist=00DDA
SPLTAS =  0E13F PM%STA        -
SPLTAX =  0E628 PM%STA        -
SQR    =  0C16F AB%FCN        - 1E05A JP%TAB(00356) Type=1.2 Nibs=5 Dist=11EEB
                              + 1EAC4 SB%TAB(0058D) Type=1.2 Nibs=5 Dist=12955
SQR12  =  0C530 JT%MTH        - 0C177 AB%FCN(00B3E) Type=1.1 Nibs=3 Dist=003B9
SQR15  =  0C534 JT%MTH        - 0D9E6 SM%MTH(005B1) Type=1.1 Nibs=4 Dist=014B2
SQR15M =  0C534 JT%MTH        - 0D62F SM%MTH(001FA) Type=1.1 Nibs=4 Dist=010FB
SQR17  =  0C553 JT%MTH        - 0DCBC SM%MTH(00887) Type=1.1 Nibs=4 Dist=01769
SQR70  =  0C5C3 JT%MTH        -
SQRSAV =  0D629 SM%MTH        - 0E312 PM%STA(004E3) Type=1.1 Nibs=4 Dist=00CE9
                              + 0E3BF PM%STA(00590) Type=1.1 Nibs=4 Dist=00D96
SRLEAS =  015EC TI%UTL        - 124F3 JP%POL(001D8) Type=0.1 Nibs=5
SST    =  07228 JP%SYS        - 004BD SB%DVR(004BD) Type=1.0 Nibs=4 Dist=06D6B
ST01   =  12AAC MN%TM:        - 08600 JP%EXC(00824) Type=0.1 Nibs=5
                              + 1C8F3 MN%CD:(0007A) Type=0.1 Nibs=5
STAB1  =  0D3D9 JT%MTH        - 0D713 SM%MTH(002DE) Type=1.0 Nibs=3 Dist=0033A
                              + 0E00D PM%STA(001DE) Type=1.1 Nibs=4 Dist=00C34
                              + 0E1DE PM%STA(003AF) Type=1.1 Nibs=4 Dist=00E05
STAB1X =  0D712 SM%MTH        - 0DEC0 PM%STA(00091) Type=1.1 Nibs=3 Dist=007AE
STAB2  =  0D400 JT%MTH        - 0DC97 SM%MTH(00862) Type=1.1 Nibs=4 Dist=00897
STAKDN =  15F75 AB%CLC        - 097F5 TI%ERD(00462) Type=0.1 Nibs=5
                              + 16366 AB%BLD(0003A) Type=1.0 Nibs=3 Dist=003F1
                              + 166F7 AB%ED:(00085) Type=1.1 Nibs=3 Dist=00782
STAKUP =  15F48 AB%CLC        - 097B6 TI%ERD(00423) Type=0.1 Nibs=5
                              + 16334 AB%BLD(00008) Type=1.1 Nibs=3 Dist=003EC
                              + 166EC AB%ED:(0007A) Type=1.1 Nibs=3 Dist=007A4
STAT   =  0B014 AB%REG        - 1ECE0 SB%TAB(007A9) Type=1.2 Nibs=5 Dist=13CCC
STATAR =  2F7AD SB%RAM        - 0B056 AB%REG(0045E) Type=0.0 Nibs=5
                              + 0E5D5 PM%STA(007A6) Type=0.0 Nibs=5
STATDC =  058F2 SG%LDC        - 0B00A AB%REG(00412) Type=1.2 Nibs=5 Dist=05718
STATIC =  0F27A AB%EXP        - 1E7EB SB%TAB(002B4) Type=1.2 Nibs=5 Dist=0F571
                              + 1E7F4 SB%TAB(002BD) Type=1.2 Nibs=5 Dist=0F57A
                              + 1E7FD SB%TAB(002C6) Type=1.2 Nibs=5 Dist=0F583
                              + 1E806 SB%TAB(002CF) Type=1.2 Nibs=5 Dist=0F58C
                              + 1E80F SB%TAB(002D8) Type=1.2 Nibs=5 Dist=0F595
                              + 1E818 SB%TAB(002E1) Type=1.2 Nibs=5 Dist=0F59E
                              + 1E821 SB%TAB(002EA) Type=1.2 Nibs=5 Dist=0F5A7
                              + 1E82A SB%TAB(002F3) Type=1.2 Nibs=5 Dist=0F5B0
                              + 1E833 SB%TAB(002FC) Type=1.2 Nibs=5 Dist=0F5B9
                              + 1E83C SB%TAB(00305) Type=1.2 Nibs=5 Dist=0F5C2
                              + 1E845 SB%TAB(0030E) Type=1.2 Nibs=5 Dist=0F5CB
                              + 1E84E SB%TAB(00317) Type=1.2 Nibs=5 Dist=0F5D4
                              + 1E857 SB%TAB(00320) Type=1.2 Nibs=5 Dist=0F5DD
                              + 1E860 SB%TAB(00329) Type=1.2 Nibs=5 Dist=0F5E6
                              + 1E869 SB%TAB(00332) Type=1.2 Nibs=5 Dist=0F5EF
                              + 1E872 SB%TAB(0033B) Type=1.2 Nibs=5 Dist=0F5F8
                              + 1E87B SB%TAB(00344) Type=1.2 Nibs=5 Dist=0F601
                              + 1E884 SB%TAB(0034D) Type=1.2 Nibs=5 Dist=0F60A
```

```
                                    + 1E88D SB%TAB(00356) Type=1.2 Nibs=5 Dist=0F613
                                    + 1E896 SB%TAB(0035F) Type=1.2 Nibs=5 Dist=0F61C
                                    + 1E89F SB%TAB(00368) Type=1.2 Nibs=5 Dist=0F625
                                    + 1E8A8 SB%TAB(00371) Type=1.2 Nibs=5 Dist=0F62E
                                    + 1E8B1 SB%TAB(0037A) Type=1.2 Nibs=5 Dist=0F637
                                    + 1E8BA SB%TAB(00383) Type=1.2 Nibs=5 Dist=0F640
                                    + 1E8C3 SB%TAB(0038C) Type=1.2 Nibs=5 Dist=0F649
                                    + 1E8CC SB%TAB(00395) Type=1.2 Nibs=5 Dist=0F652
STATP  =  03A0E JP%PR3              - 0B00F AB%REG(00417) Type=1.2 Nibs=5 Dist=07601
STATR+ =  1730C FH%TFM              - 1701C FH%TFM(0056E) Type=1.1 Nibs=3 Dist=002F0
STATRS =  172F3 FH%TFM              -
STATSV =  1732F FH%TFM              -
STCD2  =  0D427 JT%MTH              -
STD    =  0EFF7 MN%UTL              - 1DDD2 JP%TAB(000CE) Type=1.2 Nibs=5 Dist=0EDDB
STDRG? =  1CE59 MN%CD:              -
STFPTR =  113A4 SC%FIL              -
STHD10 =  14C3E SC%DAT              -
STK16? =  13DB3 SC%DAT              - 0E260 PM%STA(00431) Type=1.1 Nibs=4 Dist=05B53
STK19? =  13DB0 SC%DAT              - 19A6E SC%SUB(00CFC) Type=1.1 Nibs=4 Dist=05CBE
                                    + 19AE2 SC%SUB(00D70) Type=1.1 Nibs=4 Dist=05D32
STKBAK =  15F68 AB%CLC              - 16819 AB%ED:(001A7) Type=1.1 Nibs=4 Dist=008B1
STKCH+ =  18507 SB%IO:              -
STKCHR =  18504 SB%IO:              - 0A9B6 SB%FCN(0008B) Type=0.1 Nibs=5
                                    + 17D49 SB%RD:(00182) Type=1.1 Nibs=3 Dist=007BB
                                    + 1ACD2 SG%KEY(00231) Type=1.1 Nibs=4 Dist=027CE
                                    + 1AD9E SB%VAL(00030) Type=1.1 Nibs=4 Dist=0289A
STKCMD =  155ED AB%CLC              - 017C5 SB%CMD(0019E) Type=0.1 Nibs=5
STKEND =  1551F AB%CLC              -
STKVC+ =  14708 SC%DAT              -
STKVCT =  1470C SC%DAT              - 17C04 SB%RD:(0003D) Type=1.1 Nibs=4 Dist=034F8
                                    + 17D92 SB%RD:(001CB) Type=1.1 Nibs=4 Dist=03686
STKWP  =  1A55D SB%EXC              -
STLXP2 =  04D09 AB%LEX              - 02FC8 JP%PR2(003A2) Type=1.1 Nibs=4 Dist=01D41
STLXPT =  04D06 AB%LEX              - 03E77 JP%PR3(00AA1) Type=1.1 Nibs=4 Dist=00E8F
STMBCL =  090E7 SG%EXC              - 0039C SB%DVR(0039C) Type=0.1 Nibs=5
                                    + 071C3 JP%SYS(00294) Type=1.1 Nibs=4 Dist=01F24
STMBUF =  090DF SG%EXC              - 195B1 SC%SUB(0083F) Type=0.1 Nibs=5
                                    + 1A023 SC%SUB(012B1) Type=0.1 Nibs=5
STMEND =  12DC6 MN%TM:              - 134D7 PM%FLG(0008E) Type=1.0 Nibs=3 Dist=00711
STMTD0 =  2F891 SB%RAM              - 025F5 JP%PR1(00055) Type=0.0 Nibs=4
                                    + 02992 JP%PR1(003F2) Type=0.0 Nibs=5
                                    + 02FB8 JP%PR2(00392) Type=0.0 Nibs=5
                                    + 0702D JP%SYS(000FE) Type=0.0 Nibs=5
                                    + 0703E JP%SYS(0010F) Type=0.0 Nibs=5
                                    + 1129F SC%FIL(0024C) Type=0.0 Nibs=5
                                    + 1149F SC%FIL(0044C) Type=0.0 Nibs=5
                                    + 14B75 SC%DAT(012C0) Type=0.0 Nibs=5
                                    + 187CB SB%IO:(00A0F) Type=0.0 Nibs=5
                                    + 188A3 SB%IO:(00AE7) Type=0.0 Nibs=4
                                    + 1921A SC%SUB(004A8) Type=0.0 Nibs=5
                                    + 19480 SC%SUB(0070E) Type=0.0 Nibs=5
                                    + 19673 SC%SUB(00901) Type=0.0 Nibs=5
                                    + 19F01 SC%SUB(0118F) Type=0.0 Nibs=5
STMTD1 =  2F896 SB%RAM              - 0E1F3 PM%STA(003C4) Type=0.0 Nibs=5
                                    + 0E20D PM%STA(003DE) Type=0.0 Nibs=5
                                    + 111FB SC%FIL(001A8) Type=0.0 Nibs=5
                                    + 11367 SC%FIL(00314) Type=0.0 Nibs=5
                                    + 11394 SC%FIL(00341) Type=0.0 Nibs=5
                                    + 11486 SC%FIL(00433) Type=0.0 Nibs=5
                                    + 12209 SC%FIL(011B6) Type=0.0 Nibs=5
```

```
                                + 13AC7 SC%DAT(00212) Type=0.0 Nibs=5
                                + 13D03 SC%DAT(0044E) Type=0.0 Nibs=5
                                + 14A8E SC%DAT(011D9) Type=0.0 Nibs=5
                                + 14B03 SC%DAT(0124E) Type=0.0 Nibs=5
                                + 14BCB SC%DAT(01316) Type=0.0 Nibs=5
                                + 15988 AB%CLC(00469) Type=0.0 Nibs=5
                                + 159D4 AB%CLC(004B5) Type=0.0 Nibs=5
                                + 15B39 AB%CLC(0061A) Type=0.0 Nibs=5
                                + 16200 AB%CLC(00CE1) Type=0.0 Nibs=4
                                + 171F9 FH%TFM(0074B) Type=0.0 Nibs=5
                                + 17228 FH%TFM(0077A) Type=0.0 Nibs=5
                                + 19D35 SC%SUB(00FC3) Type=0.0 Nibs=5
STMTL+ =  02DC4 JP%PR2          - 02808 JP%PR1(00268) Type=1.1 Nibs=3 Dist=005BC
                                + 02858 JP%PR1(002B8) Type=1.1 Nibs=3 Dist=0056C
STMTLO =  02DC7 JP%PR2          - 034EF JP%PR3(00119) Type=1.1 Nibs=3 Dist=00728
STMTNF =  1B4E6 MB%IMG          - 0717D JP%SYS(0024E) Type=0.1 Nibs=5
                                + 1A5BD SC%REN(00041) Type=1.0 Nibs=4 Dist=00F29
STMTRO =  2F871 SB%RAM          - 07F7F JP%EXC(001A3) Type=0.0 Nibs=5
                                + 115D9 SC%FIL(00586) Type=0.0 Nibs=5
                                + 116A5 SC%FIL(00652) Type=0.0 Nibs=5
                                + 11B62 SC%FIL(00B0F) Type=0.0 Nibs=5
                                + 11C43 SC%FIL(00BF0) Type=0.0 Nibs=5 Offset=      7
                                + 11C94 SC%FIL(00C41) Type=0.0 Nibs=5
                                + 11CC4 SC%FIL(00C71) Type=0.0 Nibs=5
                                + 11CF9 SC%FIL(00CA6) Type=0.0 Nibs=5
                                + 11D21 SC%FIL(00CCE) Type=0.0 Nibs=5 Offset=     11
                                + 11DEF SC%FIL(00D9C) Type=0.0 Nibs=5
                                + 11E13 SC%FIL(00DC0) Type=0.0 Nibs=5
                                + 17405 FH%TFM(00957) Type=0.0 Nibs=5
                                + 17440 FH%TFM(00992) Type=0.0 Nibs=5
                                + 17DC3 SB%IO:(00007) Type=0.0 Nibs=5 Offset=     11
                                + 17E09 SB%IO:(0004D) Type=0.0 Nibs=5 Offset=      1
                                + 180A2 SB%IO:(002E6) Type=0.0 Nibs=5 Offset=      1
                                + 18103 SB%IO:(00347) Type=0.0 Nibs=5 Offset=      6
                                + 1857B SB%IO:(007BF) Type=0.0 Nibs=5 Offset=      6
STMTR1 =  2F881 SB%RAM          - 1163F SC%FIL(005EC) Type=0.0 Nibs=5 Offset=      4
                                + 11FC8 SC%FIL(00F75) Type=0.0 Nibs=5 Offset=      4
STMTST =  18B20 SB%IO:          - 0F016 MN%UTL(003A8) Type=0.1 Nibs=5
STOP   =  07696 JP%SYS          - 1ED43 SB%TAB(0080C) Type=1.2 Nibs=5 Dist=176AD
STOPDC =  05303 SG%LDC          - 0768C JP%SYS(0075D) Type=1.2 Nibs=5 Dist=02389
                                + 076C8 JP%SYS(00799) Type=1.2 Nibs=5 Dist=023C5
                                + 07FD8 JP%EXC(001FC) Type=1.2 Nibs=5 Dist=02CD5
                                + 08FC5 SG%EXC(0097D) Type=1.2 Nibs=5 Dist=03CC2
                                + 08FD6 SG%EXC(0098E) Type=1.2 Nibs=5 Dist=03CD3
                                + 0DE44 PM%STA(00015) Type=1.2 Nibs=5 Dist=08B41
                                + 0EFED MN%UTL(0037F) Type=1.2 Nibs=5 Dist=09CEA
                                + 12DCC MN%TM:(00882) Type=1.2 Nibs=5 Dist=0DAC9
                                + 13859 PM%FLG(00410) Type=1.2 Nibs=5 Dist=0E556
                                + 1386B PM%FLG(00422) Type=1.2 Nibs=5 Dist=0E568
                                + 1D70B MN%CD:(00E92) Type=1.2 Nibs=5 Dist=18408
STORE  =  0F5F8 AB%ASN          - 0892A SG%EXC(002E2) Type=1.0 Nibs=4 Dist=06CCE
                                + 0B353 AB%REG(0075B) Type=0.0 Nibs=5
                                + 13F47 SC%DAT(00692) Type=1.1 Nibs=4 Dist=0494F
                                + 14005 SC%DAT(00750) Type=0.0 Nibs=5
                                + 15E38 AB%CLC(00919) Type=1.1 Nibs=4 Dist=06840
                                + 18861 SB%IO:(00AA5) Type=0.0 Nibs=5
                                + 18BB3 SB%IO:(00DF7) Type=0.1 Nibs=5
STORTN =  14015 SC%DAT          - 13FFC SC%DAT(00747) Type=0.0 Nibs=5
STOSTA =  01BAB SB%DSP          - 1C4AA SB%GPH(000E5) Type=0.1 Nibs=5
STPASG =  11A9B SC%FIL          - 191B0 SC%SUB(0043E) Type=1.1 Nibs=4 Dist=07715
```

```
STR$    =  18156 SB%IO:        - 1EB78 SB%TAB(00641) Type=1.2 Nibs=5 Dist=06A22
STR$00  =  1815C SB%IO:        -
STR$SB  =  18149 SB%IO:        - 0FDCA SC%TRC(00383) Type=0.1 Nibs=5
                               + 14494 SC%DAT(00BDF) Type=1.1 Nibs=4 Dist=03CB5
                               + 1633F AB%BLD(00013) Type=1.1 Nibs=4 Dist=01E0A
                               + 1BCA7 MB%USG(00203) Type=1.1 Nibs=4 Dist=03B5E
STRASN  =  0F6B3 AB%ASN        - 193C1 SC%SUB(0064F) Type=0.1 Nibs=5
STREQL  =  1B1EF AB%UTL        - 0C0AA AB%FCN(00A71) Type=0.1 Nibs=5
                               + 15655 AB%CLC(00136) Type=1.1 Nibs=4 Dist=05B9A
STRGCK  =  0368A JP%PR3        - 03227 JP%PR2(00601) Type=1.1 Nibs=3 Dist=00493
STRHDR  =  0F09A MN%UTL        - 12C24 MN%TM:(006DA) Type=1.1 Nibs=4 Dist=03B8A
                               + 1C539 SG%POK(0004B) Type=0.1 Nibs=5
                               + 1C843 SG%POK(00355) Type=0.1 Nibs=5
STRHED  =  14C2E SC%DAT        -
STRING  =  0F39C AB%EXP        - 1E737 SB%TAB(00200) Type=1.2 Nibs=5 Dist=0F39B
STRLIT  =  0C1AA AB%FCN        - 1E6D4 SB%TAB(0019D) Type=1.2 Nibs=5 Dist=1252A
                               + 1E701 SB%TAB(001CA) Type=1.2 Nibs=5 Dist=12557
STRNGP  =  0379D JP%PR3        - 092BC SG%EXC(00C74) Type=1.2 Nibs=5 Dist=05B1F
                               + 18C5D MN%LCK(00005) Type=1.2 Nibs=5 Dist=154C0
                               + 1A489 SB%EXC(000B5) Type=1.2 Nibs=5 Dist=16CEC
                               + 1A4C9 SB%EXC(000F5) Type=1.2 Nibs=5 Dist=16D2C
                               + 1C45D SB%GPH(00098) Type=1.2 Nibs=5 Dist=18CC0
STROVF  =  1411A SC%DAT        - 18FA9 SC%SUB(00237) Type=1.0 Nibs=4 Dist=04E8F
                               + 1A4C0 SB%EXC(000EC) Type=1.0 Nibs=4 Dist=063A6
STRRCL  =  0F40C AB%EXP        - 0BB85 AB%FCN(0054C) Type=1.0 Nibs=4 Dist=03887
STRSPC  =  14A44 SC%DAT        -
STRTST  =  1B1C7 AB%UTL        - 0B83A AB%FCN(00201) Type=0.1 Nibs=5
STRTUP  =  1A48E SB%EXC        - 1E063 JP%TAB(0035F) Type=1.2 Nibs=5 Dist=03BD5
STSAVE  =  2F6BE SB%RAM        - 0F5BA AB%EXP(00442) Type=0.0 Nibs=5
                               + 0F5CA AB%EXP(00452) Type=0.0 Nibs=5
                               + 19CEB SC%SUB(00F79) Type=0.0 Nibs=5
                               + 19F5B SC%SUB(011E9) Type=0.0 Nibs=5
STSCR   =  0E92C PM%STA        -
STUFF   =  1B0B2 AB%UTL        - 16436 AB%BLD(0010A) Type=1.0 Nibs=4 Dist=04C7C
SUB     =  18D7C SC%SUB        - 1EC6B SB%TAB(00734) Type=1.2 Nibs=5 Dist=05EEF
SUB$    =  0C1FC AB%FCN        - 044B5 SB%EXP(004E0) Type=0.0 Nibs=5 Offset=     -2
                               + 044C4 SB%EXP(004EF) Type=0.0 Nibs=1 Offset=     -2
                               + 1EB81 SB%TAB(0064A) Type=1.2 Nibs=5 Dist=12985
SUBCHK  =  19764 SC%SUB        - 0765C JP%SYS(0072D) Type=0.1 Nibs=5
SUBDC   =  0586A SG%LDC        - 18D72 SC%SUB(00000) Type=1.2 Nibs=5 Dist=13508
SUBONE  =  0C327 JT%MTH        - 0DCA9 SM%MTH(00874) Type=1.1 Nibs=4 Dist=01982
                               + 0E870 PM%STA(00A41) Type=1.1 Nibs=4 Dist=02549
SUBP    =  037B2 JP%PR3        - 18D77 SC%SUB(00005) Type=1.2 Nibs=5 Dist=155C5
SVFTYP  =  11560 SC%FIL        -
SVINF+  =  08457 JP%EXC        - 06FF3 JP%SYS(000C4) Type=1.1 Nibs=4 Dist=01464
SVINF0  =  0845A JP%EXC        - 0125E TI%UTL(0012D) Type=1.0 Nibs=4 Dist=071FC
                               + 07001 JP%SYS(000D2) Type=1.1 Nibs=4 Dist=01459
                               + 171B4 FH%TFM(00706) Type=0.1 Nibs=5
SVRST+  =  025A6 JP%PR1        - 0349A JP%PR3(000C4) Type=1.1 Nibs=4 Dist=00EF4
SVRST2  =  025C5 JP%PR1        - 02FEA JP%PR2(003C4) Type=1.1 Nibs=4 Dist=00A25
SVTRC   =  0FA35 AB%ASN        - 0E489 PM%STA(0065A) Type=1.1 Nibs=4 Dist=015AC
                               + 13E4E SC%DAT(00599) Type=1.1 Nibs=4 Dist=04419
SWAPXY  =  0DA0B SM%MTH        - 0DF0A PM%STA(000DB) Type=1.1 Nibs=3 Dist=004FF
                               + 0DFB5 PM%STA(00186) Type=1.1 Nibs=3 Dist=005AA
SWPBYT  =  17A24 FH%TFM        - 1D64E MN%CD:(00DD5) Type=1.1 Nibs=4 Dist=05C2A
SYCOLL  =  01455 TI%UTL        - 028D0 JP%PR1(00330) Type=1.1 Nibs=4 Dist=0147B
SYNTAX  =  15D09 AB%CLC        - 16789 AB%ED:(00117) Type=1.1 Nibs=4 Dist=00A80
SYNTXe  =  02E2B JP%PR2        -
SYSEN   =  2F58A SB%RAM        - 01457 TI%UTL(00326) Type=0.0 Nibs=5
                               + 0F1E5 AB%EXP(0006D) Type=0.0 Nibs=5
```

```
SYSERR =  17B27 MN%FRP        - 1AF59 MN%GSB(00058) Type=1.0 Nibs=4 Dist=03432
SYSFLG =  2F6D9 SB%RAM        - 00705 SB%DVR(00705) Type=0.0 Nibs=5 Offset=     10
                              + 01C30 SB%DSP(003A0) Type=0.0 Nibs=5 Offset=     11
                              + 02BE9 JP%PR1(00649) Type=0.0 Nibs=5 Offset=     10
                              + 0CBFB JT%MTH(008D4) Type=0.0 Nibs=5 Offset=     -1
                              + 0F955 AB%ASN(0037B) Type=0.0 Nibs=5 Offset=      3
                              + 13507 PM%FLG(000BE) Type=0.0 Nibs=5
                              + 14759 SC%DAT(00EA4) Type=0.0 Nibs=5 Offset=      3
SYSTEM =  1554C AB%CLC        -
SavLvl =  00005 SB%DSP        -
Scopck =  076C2 JP%SYS        - 0790A JP%SYS(009DB) Type=1.1 Nibs=3 Dist=00248
SetAVE =  1BA00 MB%IMG        - 1C027 MB%USG(00583) Type=1.1 Nibs=3 Dist=00627
                              + 1C079 MB%USG(005D5) Type=1.0 Nibs=3 Dist=00679
                              + 1C11F MB%USG(0067B) Type=1.1 Nibs=3 Dist=0071F
SetAVM =  1B9FA MB%IMG        - 1C064 MB%USG(005C0) Type=1.1 Nibs=3 Dist=0066A
                              + 1C2FC MB%USG(00858) Type=1.1 Nibs=4 Dist=00902
Sflag? =  15109 MN%ED:        - 1CFEF MN%CD:(00776) Type=1.1 Nibs=4 Dist=07EE6
SflagC =  076E2 JP%SYS        - 0951F TI%ERD(0018C) Type=1.1 Nibs=4 Dist=01E3D
SflgCp =  076DE JP%SYS        -
Snaprs =  1244F JP%POL        - 1160F SC%FIL(005BC) Type=1.1 Nibs=4 Dist=00E40
                              + 11E0E SC%FIL(00DBB) Type=1.1 Nibs=3 Dist=00641

TAN    =  0BE7B AB%FCN        - 1EAFA SB%TAB(005C3) Type=1.2 Nibs=5 Dist=12C7F
TAN12  =  0D72F SM%MTH        - 0BE84 AB%FCN(0084B) Type=1.1 Nibs=4 Dist=018AB
TAN15  =  0D733 SM%MTH        -
TASTK  =  2F599 SB%RAM        -
TBLJMC =  02426 SB%DSP        - 003F7 SB%DVR(003F7) Type=1.1 Nibs=4 Dist=0202F
                              + 14D6A MN%ED:(000EA) Type=0.1 Nibs=5
TBLJMP =  0242A SB%DSP        - 1B70E MB%IMG(002C8) Type=0.1 Nibs=5
TBLU   =  0E601 PM%STA        -
TBMSG$ =  099AB TI%ERD        -
TBMSTX =  099AD TI%ERD        -
TERCHR =  2F97D SB%RAM        -
TFHDLR =  1702F FH%TFM        -
TFORN  =  2F59E SB%RAM        - 1C670 SG%POK(00182) Type=0.0 Nibs=5
TGSBS  =  2F5A3 SB%RAM        -
TIMAF  =  2F787 SB%RAM        - 129CC MN%TM:(00482) Type=0.0 Nibs=5
                              + 129F5 MN%TM:(004AB) Type=0.0 Nibs=5
TIME   =  12B8D MN%TM:        - 1E9F5 SB%TAB(004BE) Type=1.2 Nibs=5 Dist=0BE68
TIME$  =  12BFF MN%TM:        - 1EADF SB%TAB(005A8) Type=1.2 Nibs=5 Dist=0BEE0
TIMER1 =  2E3F8 SB%RAM        - 00730 SB%DVR(00730) Type=0.0 Nibs=5 Offset=      5
                              + 00CF9 SB%KEY(00002) Type=0.0 Nibs=5
                              + 15340 MN%ED:(006C0) Type=0.0 Nibs=5 Offset=      5
                              + 1538D MN%ED:(0070D) Type=0.0 Nibs=5
                              + 1D150 MN%CD:(008D7) Type=0.0 Nibs=5
                              + 1D197 MN%CD:(0091E) Type=0.0 Nibs=5 Offset=      5
TIMER2 =  2E2F8 SB%RAM        - 0EA01 MN%BP:(00010) Type=0.0 Nibs=5
                              + 1254E MN%TM:(00004) Type=0.0 Nibs=5
                              + 127DB MN%TM:(00291) Type=0.0 Nibs=5
                              + 13138 MN%TM:(00BEE) Type=0.0 Nibs=5
TIMER3 =  2E1F8 SB%RAM        - 001A8 SB%DVR(001A8) Type=0.0 Nibs=5 Offset=      7
                              + 0E65F PM%STA(00830) Type=0.0 Nibs=5
                              + 0EF1D MN%UTL(002AF) Type=0.0 Nibs=5 Offset=      5
                              + 0EF5D MN%UTL(002EF) Type=0.0 Nibs=5
TIMLAF =  2F77B SB%RAM        - 12A02 MN%TM:(004B8) Type=0.0 Nibs=5
                              + 12A1C MN%TM:(004D2) Type=0.0 Nibs=5
TIMLST =  2F76F SB%RAM        - 12A29 MN%TM:(004DF) Type=0.0 Nibs=5
                              + 12A34 MN%TM:(004EA) Type=0.0 Nibs=5
TIMOFS =  2F763 SB%RAM        - 1298C MN%TM:(00442) Type=0.0 Nibs=5
                              + 129AB MN%TM:(00461) Type=0.0 Nibs=5
```

```
TIMRND =  12611 MN%TM:          -
TKSCN+ =  08A6B SG%EXC          -
TKSCN4 =  08A71 SG%EXC          - 17C69 SB%RD:(000A2) Type=0.1 Nibs=5
TKSCN7 =  08A99 SG%EXC          - 07836 JP%SYS(00907) Type=1.1 Nibs=4 Dist=01263
TMIADR =  080E9 JP%EXC          -
TMRAD1 =  2F697 SB%RAM          - 08101 JP%EXC(00325) Type=0.0 Nibs=5 Offset=     -5
TMRAD2 =  2F69C SB%RAM          -
TMRAD3 =  2F6A1 SB%RAM          -
TMRIN1 =  2F6A6 SB%RAM          - 080EB JP%EXC(0030F) Type=0.0 Nibs=5 Offset=     -8
TMRIN2 =  2F6AE SB%RAM          -
TMRIN3 =  2F6B6 SB%RAM          -
TNOFF? =  006AF SB%DVR          - 15894 AB%CLC(00375) Type=0.1 Nibs=5
TODT   =  13229 MN%TM:          - 1C901 MN%CD:(00088) Type=0.1 Nibs=5
TONE   =  0EBEB MN%BP:          -
TOTAL  =  0E2C8 PM%STA          - 1E06C JP%TAB(00368) Type=1.2 Nibs=5 Dist=0FDA4
TRACDC =  052FC SG%LDC          - 0FA47 SC%TRC(00000) Type=1.2 Nibs=5 Dist=0A74B
TRACE  =  0FA51 SC%TRC          - 1EDDC SB%TAB(008A5) Type=1.2 Nibs=5 Dist=0F38B
TRACEA =  0FA9A SC%TRC          - 0F635 AB%ASN(0005B) Type=1.0 Nibs=3 Dist=00465
TRACEM =  2F7B0 SB%RAM          - 0FF3B SC%TRC(004F4) Type=0.0 Nibs=5
TRACEP =  03335 JP%PR2          - 0FA4C SC%TRC(00005) Type=1.2 Nibs=5 Dist=0C717
TRAP   =  136F4 PM%FLG          - 1E07E JP%TAB(0037A) Type=1.2 Nibs=5 Dist=0A98A
TRC90  =  0DA11 SM%MTH          - 0DAAD SM%MTH(00678) Type=0.0 Nibs=5 Offset=    224
TRCFI# =  173D0 FH%TFM          -
TRCLIN =  0FEC4 SC%TRC          - 190A8 SC%SUB(00336) Type=0.1 Nibs=5
                                + 195BF SC%SUB(0084D) Type=0.1 Nibs=5
TRCRTN =  0F638 AB%ASN          - 0FDFF SC%TRC(003B8) Type=1.0 Nibs=3 Dist=007C7
TRFCK- =  0FE1B SC%TRC          - 07B53 JP%SYS(00C24) Type=0.1 Nibs=5
TRFLCK =  0FE18 SC%TRC          - 08C6E SG%EXC(00626) Type=1.0 Nibs=4 Dist=071AA
                                + 198E3 SC%SUB(00B71) Type=0.1 Nibs=5
TRFM20 =  0FE6E SC%TRC          -
TRFMBF =  2F8C5 SB%RAM          - 07325 JP%SYS(003F6) Type=0.0 Nibs=5
TRFROM =  0FE59 SC%TRC          - 07A19 JP%SYS(00AEA) Type=0.1 Nibs=5
                                + 0804B JP%EXC(0026F) Type=1.1 Nibs=4 Dist=07E0E
                                + 08C60 SG%EXC(00618) Type=1.1 Nibs=4 Dist=071F9
                                + 19DBD SC%SUB(0104B) Type=0.1 Nibs=5
                                + 19ED9 SC%SUB(01167) Type=0.1 Nibs=5
TRKDON =  1CFAC MN%CD:          -
TRMNTR =  0F1DD AB%EXP          - 1E5AB SB%TAB(00074) Type=1.2 Nibs=5 Dist=0F3CE
                                + 1E617 SB%TAB(000E0) Type=1.2 Nibs=5 Dist=0F43A
                                + 1E620 SB%TAB(000E9) Type=1.2 Nibs=5 Dist=0F443
                                + 1E629 SB%TAB(000F2) Type=1.2 Nibs=5 Dist=0F44C
                                + 1E632 SB%TAB(000FB) Type=1.2 Nibs=5 Dist=0F455
                                + 1E63B SB%TAB(00104) Type=1.2 Nibs=5 Dist=0F45E
                                + 1E6B0 SB%TAB(00179) Type=1.2 Nibs=5 Dist=0F4D3
                                + 1E6B9 SB%TAB(00182) Type=1.2 Nibs=5 Dist=0F4DC
                                + 1E6C2 SB%TAB(0018B) Type=1.2 Nibs=5 Dist=0F4E5
                                + 1E6CB SB%TAB(00194) Type=1.2 Nibs=5 Dist=0F4EE
                                + 1E6DD SB%TAB(001A6) Type=1.2 Nibs=5 Dist=0F500
                                + 1E6E6 SB%TAB(001AF) Type=1.2 Nibs=5 Dist=0F509
                                + 1E6EF SB%TAB(001B8) Type=1.2 Nibs=5 Dist=0F512
                                + 1E6F8 SB%TAB(001C1) Type=1.2 Nibs=5 Dist=0F51B
                                + 1E70A SB%TAB(001D3) Type=1.2 Nibs=5 Dist=0F52D
                                + 1E713 SB%TAB(001DC) Type=1.2 Nibs=5 Dist=0F536
                                + 1E71C SB%TAB(001E5) Type=1.2 Nibs=5 Dist=0F53F
                                + 1E725 SB%TAB(001EE) Type=1.2 Nibs=5 Dist=0F548
                                + 1E72E SB%TAB(001F7) Type=1.2 Nibs=5 Dist=0F551
                                + 1E740 SB%TAB(00209) Type=1.2 Nibs=5 Dist=0F563
                                + 1E749 SB%TAB(00212) Type=1.2 Nibs=5 Dist=0F56C
                                + 1E7AC SB%TAB(00275) Type=1.2 Nibs=5 Dist=0F5CF
                                + 1E7B5 SB%TAB(0027E) Type=1.2 Nibs=5 Dist=0F5D8
```

```
                                    + 1E7BE SB%TAB(00287) Type=1.2 Nibs=5 Dist=0F5E1
                                    + 1E7C7 SB%TAB(00290) Type=1.2 Nibs=5 Dist=0F5EA
                                    + 1E7D0 SB%TAB(00299) Type=1.2 Nibs=5 Dist=0F5F3
                                    + 1E7D9 SB%TAB(002A2) Type=1.2 Nibs=5 Dist=0F5FC
                                    + 1E7E2 SB%TAB(002AB) Type=1.2 Nibs=5 Dist=0F605
                                    + 1E8D5 SB%TAB(0039E) Type=1.2 Nibs=5 Dist=0F6F8
                                    + 1E8DE SB%TAB(003A7) Type=1.2 Nibs=5 Dist=0F701
                                    + 1E8E7 SB%TAB(003B0) Type=1.2 Nibs=5 Dist=0F70A
                                    + 1E8F0 SB%TAB(003B9) Type=1.2 Nibs=5 Dist=0F713
                                    + 1E8F9 SB%TAB(003C2) Type=1.2 Nibs=5 Dist=0F71C
                                    + 1EE12 SB%TAB(008DB) Type=1.2 Nibs=5 Dist=0FC35
                                    + 1EE1B SB%TAB(008E4) Type=1.2 Nibs=5 Dist=0FC3E
                                    + 1EE24 SB%TAB(008ED) Type=1.2 Nibs=5 Dist=0FC47
                                    + 1EE36 SB%TAB(008FF) Type=1.2 Nibs=5 Dist=0FC59
TRNFCK =  02BE4 JP%PR1              - 02F35 JP%PR2(0030F) Type=1.1 Nibs=3 Dist=00351
                                    + 02F51 JP%PR2(0032B) Type=1.1 Nibs=3 Dist=0036D
                                    + 05366 SG%LDC(00408) Type=1.1 Nibs=4 Dist=02782
                                    + 0948B TI%ERD(000F8) Type=1.1 Nibs=4 Dist=068A7
TRPREG =  2F6F9 SB%RAM              - 0CB47 JT%MTH(00820) Type=0.0 Nibs=5
                                    + 0EDCE MN%UTL(00160) Type=0.0 Nibs=5
                                    + 1372D PM%FLG(002E4) Type=0.0 Nibs=5 Offset=      4
                                    + 137A8 PM%FLG(0035F) Type=0.0 Nibs=5
TRSFM+ =  16B81 FH%TFM              -
TRSFMD =  05621 SG%LDC              - 16AAE FH%TFM(00000) Type=1.2 Nibs=5 Dist=1148D
TRSFMP =  03E0B JP%PR3              - 16AB3 FH%TFM(00005) Type=1.2 Nibs=5 Dist=12CA8
TRSFMX =  16AB8 FH%TFM              - 1E075 JP%TAB(00371) Type=1.2 Nibs=5 Dist=075BD
TRSFMu =  16B84 FH%TFM              -
TRTO*  =  0FEA9 SC%TRC              - 08062 JP%EXC(00286) Type=1.1 Nibs=4 Dist=07E47
                                    + 19DCF SC%SUB(0105D) Type=0.1 Nibs=5
TRTO+  =  0FE7B SC%TRC              - 07AF9 JP%SYS(00BCA) Type=0.1 Nibs=5
                                    + 088DC SG%EXC(00294) Type=1.1 Nibs=4 Dist=0759F
TRTO-  =  0FE8F SC%TRC              - 08812 SG%EXC(001CA) Type=1.1 Nibs=4 Dist=0767D
                                    + 1A05B SC%SUB(012E9) Type=0.1 Nibs=5
TRTOEN =  0FEBC SC%TRC              - 190D3 SC%SUB(00361) Type=0.1 Nibs=5
                                    + 195DD SC%SUB(0086B) Type=0.1 Nibs=5
TRTOr  =  0FE8C SC%TRC              - 090B6 SG%EXC(00A6E) Type=1.1 Nibs=4 Dist=06DD6
TRUNCC =  12B4A MN%TM:              - 0EEF9 MN%UTL(0028B) Type=1.1 Nibs=4 Dist=03C51
TST12A =  0D476 SM%MTH              - 08972 SG%EXC(0032A) Type=1.1 Nibs=4 Dist=04B04
                                    + 09EC5 SG%FXQ(0034F) Type=1.1 Nibs=4 Dist=035B1
                                    + 0E8D6 PM%STA(00AA7) Type=1.1 Nibs=4 Dist=01460
TST15  =  0D47A SM%MTH              -
TWO*   =  0DB38 SM%MTH              -
TWONaN =  0C6A5 JT%MTH              - 0D56F SM%MTH(0013A) Type=1.1 Nibs=4 Dist=00ECA
TXT->B =  176DD FH%TFM              -
Trace  =  0000F TI%EQU              -
TrfCk- =  07B51 JP%SYS              - 08043 JP%EXC(00267) Type=1.1 Nibs=3 Dist=004F2
                                    + 0805A JP%EXC(0027E) Type=1.1 Nibs=3 Dist=00509
TstEnd =  1C0FF MB%USG              -

UNF    =  1388C PM%FLG              - 1EBD2 SB%TAB(0069B) Type=1.2 Nibs=5 Dist=0B346
UNFNIB =  2F6FA SB%RAM              - 04627 SB%EXP(00652) Type=0.0 Nibs=5
UNP    =  00001 JT%MTH              - 046B8 SB%EXP(006E3) Type=0.0 Nibs=1
UNPRDC =  05902 SG%LDC              - 1D71E MN%CD:(00EA5) Type=1.2 Nibs=5 Dist=17E1C
UNPROT =  1D728 MN%CD:              - 1E087 JP%TAB(00383) Type=1.2 Nibs=5 Dist=0095F
UNPRTP =  02BB0 JP%PR1              - 1D723 MN%CD:(00EAA) Type=1.2 Nibs=5 Dist=1AB73
UNSECR =  0A390 SG%FXQ              - 1E090 JP%TAB(0038C) Type=1.2 Nibs=5 Dist=13D00
UPCPOS =  13C67 SC%DAT              - 1764F FH%TFM(00BA1) Type=1.1 Nibs=4 Dist=039E8
UPD1EN =  2F599 SB%RAM              -
UPD1ST =  2F55D SB%RAM              - 0A699 SG%FXQ(00B23) Type=0.0 Nibs=5
UPD2EN =  2F6A6 SB%RAM              -
```

```
UPD2ST =  2F674 SB%RAM        - 0A820 SG%FXQ(00CAA) Type=0.0 Nibs=5
UPDANN =  13571 PM%FLG        - 0EDF0 MN%UTL(00182) Type=1.0 Nibs=4 Dist=04781
UPDANX =  1356D PM%FLG        -
UPDCRL =  096E7 TI%ERD        - 1A5B6 SC%REN(0003A) Type=0.1 Nibs=5
UPDFC+ =  01617 TI%UTL        -
UPDFCL =  01614 TI%UTL        - 08C40 SG%EXC(005F8) Type=1.1 Nibs=4 Dist=0762C
UPDIN+ =  02DB2 JP%PR2        - 0280C JP%PR1(0026C) Type=1.1 Nibs=3 Dist=005A6
                              + 034F3 JP%PR3(0011D) Type=1.1 Nibs=3 Dist=00741
UPDINA =  02DAF JP%PR2        - 025A2 JP%PR1(00002) Type=1.1 Nibs=4 Dist=0080D
                              + 0294B JP%PR1(003AB) Type=1.1 Nibs=3 Dist=00464
UPDPC  =  07B58 JP%SYS        - 08056 JP%EXC(0027A) Type=1.1 Nibs=3 Dist=004FE
UPDPCA =  0AAB3 TI%FX2        - 090A6 SG%EXC(00A5E) Type=1.1 Nibs=4 Dist=01A0D
UPDPCC =  07B65 JP%SYS        - 091D9 SG%EXC(00B91) Type=1.1 Nibs=4 Dist=01674
                              + 0AACB TI%FX2(00046) Type=1.0 Nibs=4 Dist=02F66
UPRC$  =  0AA4D SB%FCN        - 1EBA5 SB%TAB(0066E) Type=1.2 Nibs=5 Dist=14158
USER   =  07DF8 JP%EXC        - 1ED94 SB%TAB(0085D) Type=1.2 Nibs=5 Dist=16F9C
USERDC =  0584C SG%LDC        - 07DEE JP%EXC(00012) Type=1.2 Nibs=5 Dist=025A2
USERK  =  1524D MN%ED:        -
USERP  =  03D9A JP%PR3        - 07DF3 JP%EXC(00017) Type=1.2 Nibs=5 Dist=04059
USERX  =  1528D MN%ED:        -
USGch+ =  1BC15 MB%USG        -
USGch- =  1BC0B MB%USG        -
USGrst =  1BC63 MB%USG        -
USING  =  1B446 MB%IMG        - 17F49 SB%ID:(0018D) Type=1.0 Nibs=4 Dist=034FD
USINGp =  03628 JP%PR3        -
USRSTA =  01B80 SB%DSP        - 00770 SB%DVR(00770) Type=1.0 Nibs=4 Dist=01410
                              + 07533 JP%SYS(00604) Type=1.1 Nibs=4 Dist=059B3
                              + 14FDB MN%ED:(0035B) Type=0.1 Nibs=5
USloop =  1C14B MB%USG        -
USnm05 =  1BD12 MB%USG        -
USst03 =  1BBCE MB%USG        -
USst05 =  1BBD4 MB%USG        -

VAL    =  1AD82 SB%VAL        - 1EB6F SB%TAB(00638) Type=1.2 Nibs=5 Dist=03DED
VALOO  =  1AD8F SB%VAL        - 14011 SC%DAT(0075C) Type=1.0 Nibs=4 Dist=06D7E
VALCHK =  1AE61 SB%VAL        - 0B362 AB%REG(0076A) Type=0.0 Nibs=5
VARDC  =  0537C SG%LDC        - 05940 SB%EXD(0001E) Type=1.1 Nibs=3 Dist=005C4
VARDC+ =  05379 SG%LDC        - 0FB2C SC%TRC(000E5) Type=0.1 Nibs=5
VARNB- =  0E28D PM%STA        -
VARNBR =  0E289 PM%STA        -
VARP   =  0350E JP%PR3        - 03290 JP%PR2(0066A) Type=1.1 Nibs=3 Dist=0027E
                              + 032BF JP%PR2(00699) Type=1.1 Nibs=3 Dist=0024F
                              + 0330C JP%PR2(006E6) Type=1.1 Nibs=3 Dist=00202
VARP05 =  03512 JP%PR3        -
VECTOR =  2F43C SB%RAM        - 00207 SB%DVR(00207) Type=0.0 Nibs=2
VECTR! =  00205 SB%DVR        -
VER$   =  0A93B SB%FCN        - 1DE98 JP%TAB(00194) Type=1.2 Nibs=5 Dist=1355D
VIEWD1 =  15147 MN%ED:        - 0017A SB%DVR(0017A) Type=0.1 Nibs=5
                              + 097E5 TI%ERD(00452) Type=0.1 Nibs=5
VIEWK  =  1529F MN%ED:        -
VRIABL =  04BC4 AB%LEX        -
ValSub =  0000A SB%VAL        - 1400E SC%DAT(00759) Type=0.0 Nibs=1

WAIT   =  0ED58 MN%UTL        - 1ED3A SB%TAB(00803) Type=1.2 Nibs=5 Dist=0FFE2
WAITKY =  10330 MN%CNF        - 1517F MN%ED:(004FF) Type=1.0 Nibs=4 Dist=04E4F
WAITP  =  02A6E JP%PR1        - 0ED53 MN%UTL(000E5) Type=1.2 Nibs=5 Dist=0C2E5
WAITd  =  05493 SG%LDC        - 0ED4E MN%UTL(000E0) Type=1.2 Nibs=5 Dist=0988B
WAKEUP =  0037E SB%DVR        -
WARM1X =  010A6 SB%KEY        - 0018C SB%DVR(0018C) Type=1.0 Nibs=4 Dist=00F1A
WARMST =  00FF3 SB%KEY        -
```

```
WFTMD- =  085D6 JP%EXC        - 083D9 JP%EXC(005FD) Type=1.1 Nibs=3 Dist=001FD
WFTMDT =  085DD JP%EXC        -
WIDTH  =  0EFA3 MN%UTL        - 1DF43 JP%TAB(0023F) Type=1.2 Nibs=5 Dist=0EFA0
WINDLN =  2F473 SB%RAM        - 018BA SB%DSP(0002A) Type=0.0 Nibs=2
                              + 01B09 SB%DSP(00279) Type=0.0 Nibs=5
                              + 01D00 SB%DSP(00470) Type=0.0 Nibs=4
                              + 01D3C SB%DSP(004AC) Type=0.0 Nibs=5
                              + 02399 SB%DSP(00B09) Type=0.0 Nibs=2
                              + 0EE55 MN%UTL(001E7) Type=0.0 Nibs=4
                              + 15149 MN%ED:(004C9) Type=0.0 Nibs=5
WINDOW =  0EDFE MN%UTL        - 1E024 JP%TAB(00320) Type=1.2 Nibs=5 Dist=0F226
WINDST =  2F471 SB%RAM        - 0245D SB%DSP(00BCD) Type=0.0 Nibs=4
                              + 0EE4B MN%UTL(001DD) Type=0.0 Nibs=5
                              + 15159 MN%ED:(004D9) Type=0.0 Nibs=2
WINDWd =  05493 SG%LDC        - 0EDF4 MN%UTL(00186) Type=1.2 Nibs=5 Dist=09961
WINDWp =  02AC6 JP%PR1        - 0EDF9 MN%UTL(0018B) Type=1.2 Nibs=5 Dist=0C333
WIPOUT =  1B0AF AB%UTL        - 00212 SB%DVR(00212) Type=0.1 Nibs=5
                              + 0026A SB%DVR(0026A) Type=0.1 Nibs=5
                              + 0B001 AB%REG(00409) Type=0.1 Nibs=5
                              + 0B518 AB%REG(00920) Type=0.1 Nibs=5
                              + 10E86 MN%CNF(00CF4) Type=0.1 Nibs=5
                              + 117FE SC%FIL(007AB) Type=0.1 Nibs=5
                              + 11C2D SC%FIL(00BDA) Type=0.1 Nibs=5
                              + 13132 MN%TM:(00BE8) Type=1.1 Nibs=4 Dist=07F7D
                              + 14ED7 MN%ED:(00257) Type=1.1 Nibs=4 Dist=061D8
                              + 16A60 AB%ED:(003EE) Type=1.1 Nibs=4 Dist=0464F
                              + 17B13 MN%FRP(000E0) Type=1.1 Nibs=4 Dist=0359C
                              + 1D927 MN%CD:(010AE) Type=1.0 Nibs=4 Dist=02878
WRBYTC =  13A73 SC%DAT        -
WRBYTD =  13A6D SC%DAT        - 17612 FH%TFM(00B64) Type=1.1 Nibs=4 Dist=03BA5
WRDSC+ =  02C26 JP%PR2        - 02AB3 JP%PR1(00513) Type=1.1 Nibs=3 Dist=00173
                              + 02B7C JP%PR1(005DC) Type=1.1 Nibs=3 Dist=000AA
                              + 03447 JP%PR3(00071) Type=1.1 Nibs=4 Dist=00821
                              + 0347F JP%PR3(000A9) Type=1.1 Nibs=4 Dist=00859
WRDSCN =  02C2A JP%PR2        - 027EB JP%PR1(0024B) Type=1.1 Nibs=3 Dist=0043F
                              + 02A7F JP%PR1(004DF) Type=1.1 Nibs=3 Dist=001AB
                              + 02B29 JP%PR1(00589) Type=1.1 Nibs=3 Dist=00101
                              + 03319 JP%PR2(006F3) Type=1.1 Nibs=3 Dist=006EF
                              + 03336 JP%PR2(00710) Type=1.1 Nibs=3 Dist=0070C
                              + 0388B JP%PR3(004B5) Type=1.0 Nibs=4 Dist=00C61
WRITNB =  1752B FH%TFM        -
WRKFIL =  0A5E3 SG%FXQ        -
WRNMST =  1FAFF TI%ERM        - 09568 TI%ERD(001D5) Type=0.0 Nibs=4 Offset=     13
WRSTR* =  13973 SC%DAT        -
WRTFIB =  11CEE SC%FIL        -
WRTNUM =  139C4 SC%DAT        -
WRTSTR =  1396F SC%DAT        -
WRTTM1 =  1538B MN%ED:        - 01B66 SB%DSP(002D6) Type=0.1 Nibs=5
WRTTMR =  15392 MN%ED:        - 001C1 SB%DVR(001C1) Type=0.1 Nibs=5
                              + 0EF72 MN%UTL(00304) Type=1.0 Nibs=4 Dist=06420
                              + 1314E MN%TM:(00C04) Type=1.1 Nibs=4 Dist=02244
WSRO-3 =  063A0 SG%SYS        - 02641 JP%PR1(000A1) Type=1.1 Nibs=4 Dist=03D5F
                              + 03472 JP%PR3(0009C) Type=1.1 Nibs=4 Dist=02F2E
WSTRFX =  138B5 SC%DAT        -

X/Y15  =  0C34F JT%MTH        - 0DCC8 SM%MTH(00893) Type=1.1 Nibs=4 Dist=01979
XCHECK =  02CC4 JP%PR2        - 029C4 JP%PR1(00424) Type=1.1 Nibs=3 Dist=00300
XDELAY =  02281 SB%DSP        - 095FA TI%ERD(00267) Type=1.1 Nibs=4 Dist=07379
XDelay =  00009 SB%DSP        -
XFN    =  0C2E5 AB%FCN        - 1EBED SB%TAB(006B6) Type=1.2 Nibs=5 Dist=12908
```

```
                                    + 1EBF6 SB%TAB(006BF) Type=1.2 Nibs=5 Dist=12911
XMTADR = 08133 JP%EXC               - 050A0 SG%LDC(00142) Type=1.1 Nibs=4 Dist=03093
                                    + 0811A JP%EXC(0033E) Type=1.1 Nibs=3 Dist=00019
                                    + 0C2EA AB%FCN(00CB1) Type=1.1 Nibs=4 Dist=041B7
XROM01 = 00001 TI%EQU               -
XTOKCK = 029C0 JP%PR1               -
XWORD  = 08116 JP%EXC               - 1EE09 SB%TAB(008D2) Type=1.2 Nibs=5 Dist=16CF3
XXHEAD = 1A44E SB%EXC               - 0A94E SB%FCN(00023) Type=0.1 Nibs=5
                                    + 18C16 SB%IO:(00E5A) Type=1.1 Nibs=4 Dist=01838
                                    + 1AD94 SB%VAL(00026) Type=1.1 Nibs=4 Dist=00946
XYEX   = 0C697 JT%MTH               - 0DA0D SM%MTH(005D8) Type=1.0 Nibs=4 Dist=01376
                                    + 0E8E4 PM%STA(00AB5) Type=1.1 Nibs=4 Dist=0224D

YMDDAY = 13304 MN%TM:               - 1C91E MN%CD:(000A5) Type=0.1 Nibs=5
YMDH01 = 130E5 MN%TM:               - 1D8B4 MN%CD:(0103B) Type=0.1 Nibs=5
YMDHMS = 130DB MN%TM:               - 08607 JP%EXC(0082B) Type=0.1 Nibs=5
YX050  = 0D2B2 JT%MTH               -
YX060  = 0D322 JT%MTH               -
YX2-12 = 0D274 JT%MTH               - 0B646 AB%FCN(0000D) Type=1.1 Nibs=4 Dist=01C2E
YX2-15 = 0D27A JT%MTH               -

ZERBUF = 18B20 SB%IO:               - 186F6 SB%IO:(0093A) Type=0.0 Nibs=5
ZERPGM = 07D54 JP%SYS               - 06D85 SG%SYS(009E5) Type=1.1 Nibs=4 Dist=00FCF
                                    + 09154 SG%EXC(00B0C) Type=1.1 Nibs=4 Dist=01400
                                    + 0A20F SG%FXQ(00699) Type=1.1 Nibs=4 Dist=024BB
                                    + 16D50 FH%TFM(002A2) Type=0.1 Nibs=5

a!     = 00021 SB%TAB               -
a"     = 00022 SB%TAB               - 036E4 JP%PR3(0030E) Type=0.0 Nibs=2
                                    + 1A9B6 SC%REN(0043A) Type=0.0 Nibs=2
                                    + 1A9F4 SC%REN(00478) Type=0.0 Nibs=2
a$     = 00024 SB%TAB               - 05FD2 SB%EXD(006B0) Type=0.0 Nibs=1
a'     = 00027 SB%TAB               - 026DA JP%PR1(0013A) Type=0.0 Nibs=1
                                    + 0358C JP%PR3(001B6) Type=0.0 Nibs=1
                                    + 036ED JP%PR3(00317) Type=0.0 Nibs=1
                                    + 1A9FD SC%REN(00481) Type=0.0 Nibs=2
a-mult = 117E6 SC%FIL               - 0AD1C AB%REG(00124) Type=1.1 Nibs=4 Dist=06ACA
                                    + 0ADA3 AB%REG(001AB) Type=1.1 Nibs=4 Dist=06A43
                                    + 0B613 AB%REG(00A1B) Type=1.1 Nibs=4 Dist=061D3
                                    + 0B62D AB%REG(00A35) Type=1.1 Nibs=4 Dist=061B9
                                    + 0B994 AB%FCN(0035B) Type=1.1 Nibs=4 Dist=05E52
                                    + 0BB78 AB%FCN(0053F) Type=1.1 Nibs=4 Dist=05C6E
                                    + 0BEAA AB%FCN(00871) Type=1.1 Nibs=4 Dist=0593C
a.     = 0002E SB%TAB               -
a0     = 00030 SB%TAB               - 05E86 SB%EXD(00564) Type=0.0 Nibs=2
a1     = 00031 SB%TAB               -
a2     = 00032 SB%TAB               -
a3     = 00033 SB%TAB               -
a4     = 00034 SB%TAB               -
a5     = 00035 SB%TAB               -
a6     = 00036 SB%TAB               -
a7     = 00037 SB%TAB               -
a8     = 00038 SB%TAB               -
a9     = 00039 SB%TAB               -
aRANGE = 150FD MN%ED:               -
atnclr = 18C51 SB%IO:               - 15901 AB%CLC(003E2) Type=1.1 Nibs=4 Dist=03350
                                    + 16706 AB%ED:(00094) Type=1.1 Nibs=4 Dist=0254B
                                    + 18CF1 MN%LCK(00099) Type=1.1 Nibs=3 Dist=000A0
                                    + 1CFDA MN%CD:(00761) Type=1.1 Nibs=4 Dist=04389
ave=d1 = 09F44 SG%FXQ               - 0667F SG%SYS(002DF) Type=1.1 Nibs=4 Dist=038C5
```

```
                                    + 066AA SG%SYS(0030A) Type=1.1 Nibs=4 Dist=0389A
                                    + 0E078 PM%STA(00249) Type=1.1 Nibs=4 Dist=04134
                                    + 0E472 PM%STA(00643) Type=1.1 Nibs=4 Dist=0452E
                                    + 0FDD1 SC%TRC(0038A) Type=1.1 Nibs=4 Dist=05E8D
                                    + 0FDF6 SC%TRC(003AF) Type=1.1 Nibs=4 Dist=05EB2

bALTCH =  00BFB TI%EQU              - 019D1 SB%DSP(00141) Type=0.0 Nibs=3
                                    + 0AA06 SB%FCN(000DB) Type=0.0 Nibs=3
bASSGN =  00804 TI%EQU              - 10ED3 MN%CNF(00D41) Type=0.0 Nibs=3
                                    + 118AD SC%FIL(0085A) Type=0.0 Nibs=3
                                    + 11AA4 SC%FIL(00A51) Type=0.0 Nibs=3
                                    + 191A2 SC%SUB(00430) Type=0.0 Nibs=3
bCARD  =  00807 TI%EQU              - 1CEF9 MN%CD:(00680) Type=0.0 Nibs=3
                                    + 1CF07 MN%CD:(0068E) Type=0.0 Nibs=3
bCHARS =  00BFB TI%EQU              - 10EE3 MN%CNF(00D51) Type=0.0 Nibs=3 Offset=  -2048
                                    + 10EFB MN%CNF(00D69) Type=0.0 Nibs=3
                                    + 1A4EB SB%EXC(00117) Type=0.0 Nibs=3
bECOMD =  00809 TI%EQU              - 00538 SB%DVR(00538) Type=0.0 Nibs=3
                                    + 005ED SB%DVR(005ED) Type=0.0 Nibs=3
                                    + 10ECA MN%CNF(00D38) Type=0.0 Nibs=3
bFIB   =  00803 TI%EQU              - 002C7 SB%DVR(002C7) Type=0.0 Nibs=3
                                    + 10ED0 MN%CNF(00D3E) Type=0.0 Nibs=3
                                    + 118A4 SC%FIL(00851) Type=0.0 Nibs=3
                                    + 11BDA SC%FIL(00B87) Type=0.0 Nibs=3
                                    + 120DB SC%FIL(01088) Type=0.0 Nibs=3
bFILE  =  00805 TI%EQU              -
bIEXKY =  00802 TI%EQU              - 003A3 SB%DVR(003A3) Type=0.0 Nibs=3
                                    + 00443 SB%DVR(00443) Type=0.0 Nibs=3
bLEX   =  00BFC TI%EQU              - 04A2F AB%LEX(00149) Type=0.0 Nibs=3
                                    + 097A1 TI%ERD(0040E) Type=0.0 Nibs=3
                                    + 10DEB MN%CNF(00C59) Type=0.0 Nibs=3
                                    + 10E22 MN%CNF(00C90) Type=0.0 Nibs=3
                                    + 10ECD MN%CNF(00D3B) Type=0.0 Nibs=3
                                    + 10FAF MN%CNF(00E1D) Type=0.0 Nibs=3
bPILAI =  00810 TI%EQU              -
bPILSV =  0080F TI%EQU              -
bROMTB =  00BFE TI%EQU              - 1C6F2 SG%POK(00204) Type=0.0 Nibs=3
bSCRTC =  00E00 TI%EQU              - 11890 SC%FIL(0083D) Type=0.0 Nibs=3
bSTART =  00808 TI%EQU              - 0054E SB%DVR(0054E) Type=0.0 Nibs=3
                                    + 10EC7 MN%CNF(00D35) Type=0.0 Nibs=3
                                    + 1A4AC SB%EXC(000D8) Type=0.0 Nibs=3
bSTAT  =  00806 TI%EQU              - 0E03E PM%STA(0020F) Type=0.0 Nibs=3
                                    + 0E09A PM%STA(0026B) Type=0.0 Nibs=3
bSTMT  =  00801 TI%EQU              - 028A4 JP%PR1(00304) Type=0.0 Nibs=3
                                    + 08FBE SG%EXC(00976) Type=0.0 Nibs=3
                                    + 10DD2 MN%CNF(00C40) Type=0.0 Nibs=3
                                    + 10EC4 MN%CNF(00D32) Type=0.0 Nibs=3
bSTMXQ =  00811 TI%EQU              -
bf2dsp =  15236 MN%ED:              - 18CDA MN%LCK(00082) Type=1.1 Nibs=4 Dist=03AA4
bf2st+ =  09818 TI%ERD              - 0933B SG%EXC(00CF3) Type=1.0 Nibs=3 Dist=004DD
bldbit =  0011D SB%DVR              -
blddsp =  00124 SB%DVR              -
bldlcd =  0012B SB%DVR              -

cC->C  =  00068 TI%EQU              - 0F69F AB%ASN(000C5) Type=0.0 Nibs=2
cR->C  =  00069 TI%EQU              - 0F68A AB%ASN(000B0) Type=0.0 Nibs=2
cRCL   =  00067 TI%EQU              - 0BBA8 AB%FCN(0056F) Type=0.0 Nibs=2
caACTV =  00042 TI%EQU              -
caCALS =  00047 TI%EQU              -
caCNTA =  0001F TI%EQU              -
```

```
caCURE =  00015 TI%EQU        -
caCURS =  00006 TI%EQU        -
caDATP =  00033 TI%EQU        -
caERRA =  00029 TI%EQU        -
caERRS =  00024 TI%EQU        -
caFSTK =  00038 TI%EQU        -
caGSTK =  0003D TI%EQU        -
caINTR =  0002E TI%EQU        -
caLENG =  00054 TI%EQU        -
caPCAD =  0001A TI%EQU        -
caPMCN =  0004C TI%EQU        -
caPMTR =  0004E TI%EQU        -
caPRME =  00010 TI%EQU        -
caPRMS =  0000B TI%EQU        -
caRTNT =  00053 TI%EQU        -
chedit =  00132 SB%DVR        -
chkspc =  0F944 AB%ASN        - 0AE8B AB%REG(00293) Type=1.1 Nibs=4 Dist=04AB9
                              + 0AF6A AB%REG(00372) Type=1.1 Nibs=4 Dist=049DA
                              + 11BEC SC%FIL(00B99) Type=1.1 Nibs=4 Dist=022A8
                              + 14AEC SC%DAT(01237) Type=1.1 Nibs=4 Dist=051A8
collap =  17F4D SB%IO:        - 17C23 SB%RD:(0005C) Type=1.1 Nibs=3 Dist=0032A
                              + 17F58 SB%IO:(0019C) Type=1.1 Nibs=3 Dist=0000B
crlfnd =  1CECE MN%CD:        - 18CCD MN%LCK(00075) Type=1.1 Nibs=4 Dist=04201
crlfsd =  0FE02 SC%TRC        - 10157 JP%MEM(001F8) Type=1.1 Nibs=3 Dist=00355
csrc5  =  0F4C8 AB%EXP        - 0A731 SG%FXQ(00BBB) Type=1.1 Nibs=4 Dist=04D97
                              + 0BA72 AB%FCN(00439) Type=1.1 Nibs=4 Dist=03A56
                              + 1179A SC%FIL(00747) Type=1.1 Nibs=4 Dist=022D2
                              + 11F64 SC%FIL(00F11) Type=1.1 Nibs=4 Dist=02A9C
csrw5  =  1D204 MN%CD:        - 1C33D MB%USG(00899) Type=1.1 Nibs=4 Dist=00EC7
cursfl =  0168C SB%CMD        - 072D6 JP%SYS(003A7) Type=1.1 Nibs=4 Dist=05C4A

d0=obs =  1725F FH%TFM        - 10111 JP%MEM(001B2) Type=1.1 Nibs=4 Dist=0714E
dCARD  =  00007 TI%EQU        - 06F9A JP%SYS(0006B) Type=0.0 Nibs=1
dIRAM  =  00001 TI%EQU        - 0834E JP%EXC(00572) Type=0.0 Nibs=1
dMAIN  =  00000 TI%EQU        -
dPCRD  =  00007 TI%EQU        -
dPORT  =  00001 TI%EQU        -
dcplin =  073E4 JP%SYS        - 06A21 SG%SYS(00681) Type=1.0 Nibs=4 Dist=009C3
debnce =  00139 SB%DVR        -
dest   =  18F3D SC%SUB        - 18C49 SB%IO:(00E8D) Type=1.1 Nibs=3 Dist=002F4
dspcha =  00140 SB%DVR        -
dsprst =  00147 SB%DVR        -
dspupd =  0014E SB%DVR        -

e#of#  =  000F7 TI%ERM        - 1FB26 TI%ERM(00746) Type=0.0 Nibs=2
                              + 1FBC5 TI%ERM(007E5) Type=0.0 Nibs=2
e0^0   =  00006 TI%ERM        - 0D3A9 JT%MTH(01082) Type=0.0 Nibs=2
e0^NEG =  00005 TI%ERM        - 0D3B6 JT%MTH(0108F) Type=0.0 Nibs=2
e1^INF =  00011 TI%ERM        - 0D310 JT%MTH(00FE9) Type=0.0 Nibs=2
e2MROM =  0001A TI%ERM        - 10EA0 MN%CNF(00D0E) Type=0.0 Nibs=4
eAF    =  0001B TI%ERM        - 12DE1 MN%TM:(00897) Type=0.0 Nibs=2
eALGN  =  000F0 TI%ERM        - 1FB48 TI%ERM(00768) Type=0.0 Nibs=2
                              + 1FB57 TI%ERM(00777) Type=0.0 Nibs=2
                              + 1FB68 TI%ERM(00788) Type=0.0 Nibs=2
                              + 1FB79 TI%ERM(00799) Type=0.0 Nibs=2
                              + 1FB8A TI%ERM(007AA) Type=0.0 Nibs=2
                              + 1FB99 TI%ERM(007B9) Type=0.0 Nibs=2
eCALGN =  00060 TI%ERM        - 1D7EE MN%CD:(00F75) Type=0.0 Nibs=2
eCHNL# =  00029 TI%ERM        - 11469 SC%FIL(00416) Type=0.0 Nibs=2
eCNTXT =  000E9 TI%ERM        - 1F802 TI%ERM(00422) Type=0.0 Nibs=2
```

```
                                + 1FA53 TI%ERM(00673) Type=0.0 Nibs=2
eDATTY =  0001F TI%ERM          - 0F94D AB%ASN(00373) Type=0.0 Nibs=2
                                + 17CCA SB%RD:(00103) Type=0.0 Nibs=2
eDVCNF =  00040 TI%ERM          - 011C8 TI%UTL(00097) Type=0.0 Nibs=4
                                + 09FA1 SG%FXQ(0042B) Type=0.0 Nibs=4
                                + 0A45B SG%FXQ(008E5) Type=0.0 Nibs=2
                                + 0BF89 AB%FCN(00950) Type=0.0 Nibs=2
                                + 1223D SC%FIL(011EA) Type=0.0 Nibs=4
                                + 1CE9F MN%CD:(00626) Type=0.0 Nibs=2
eEOFIL =  00036 TI%ERM          - 11279 SC%FIL(00226) Type=0.0 Nibs=2
                                + 13FF2 SC%DAT(0073D) Type=0.0 Nibs=2
                                + 14210 SC%DAT(0095B) Type=0.0 Nibs=2
                                + 14443 SC%DAT(00B8E) Type=0.0 Nibs=2
                                + 17259 FH%TFM(007AB) Type=0.0 Nibs=4
eEXCHR =  0004E TI%ERM          - 02E83 JP%PR2(0025D) Type=0.0 Nibs=4
                                + 02EF0 JP%PR2(002CA) Type=0.0 Nibs=4
eEXPO  =  00003 TI%ERM          - 0D5F1 SM%MTH(001BC) Type=0.0 Nibs=2
eEXPCT =  000E7 TI%ERM          - 1FA16 TI%ERM(00636) Type=0.0 Nibs=2
                                + 1FA29 TI%ERM(00649) Type=0.0 Nibs=2
                                + 1FAE3 TI%ERM(00703) Type=0.0 Nibs=2
                                + 1FAFC TI%ERM(0071C) Type=0.0 Nibs=2
eF2BIG =  0004A TI%ERM          - 1C9D8 MN%CD:(0015F) Type=0.0 Nibs=2
eFACCS =  0003C TI%ERM          - 0A11A SG%FXQ(005A4) Type=0.0 Nibs=4
                                + 142FB SC%DAT(00A46) Type=0.0 Nibs=2
                                + 16C85 FH%TFM(001D7) Type=0.0 Nibs=4
                                + 1A76D SC%REN(001F1) Type=0.0 Nibs=2
                                + 1C778 SG%POK(0028A) Type=0.0 Nibs=2
eFEXST =  0003B TI%ERM          - 08361 JP%EXC(00585) Type=0.0 Nibs=4
                                + 0A31C SG%FXQ(007A6) Type=0.0 Nibs=2
                                + 16DB1 FH%TFM(00303) Type=0.0 Nibs=4
                                + 1D3FF MN%CD:(00B86) Type=0.0 Nibs=2
eFILE  =  000EA TI%ERM          - 1F850 TI%ERM(00470) Type=0.0 Nibs=2
                                + 1F87A TI%ERM(0049A) Type=0.0 Nibs=2
                                + 1F888 TI%ERM(004A8) Type=0.0 Nibs=2
                                + 1F899 TI%ERM(004B9) Type=0.0 Nibs=2
                                + 1F8C5 TI%ERM(004E5) Type=0.0 Nibs=2
                                + 1F8D0 TI%ERM(004F0) Type=0.0 Nibs=2
                                + 1F8E6 TI%ERM(00506) Type=0.0 Nibs=2
                                + 1F955 TI%ERM(00575) Type=0.0 Nibs=2
                                + 1F9EC TI%ERM(0060C) Type=0.0 Nibs=2
eFNNtF =  00021 TI%ERM          - 15AC0 AB%CLC(005A1) Type=0.0 Nibs=2
                                + 1999F SC%SUB(00C2D) Type=0.0 Nibs=2
                                + 1F6A7 TI%ERM(002C7) Type=0.0 Nibs=2
eFOPEN =  0003E TI%ERM          - 11B94 SC%FIL(00B41) Type=0.0 Nibs=4
eFPROT =  0003D TI%ERM          - 06C18 SG%SYS(00878) Type=0.0 Nibs=4
                                + 142E1 SC%DAT(00A2C) Type=0.0 Nibs=2
                                + 1738E FH%TFM(008E0) Type=0.0 Nibs=4
                                + 1C7B1 SG%POK(002C3) Type=0.0 Nibs=2
                                + 1D439 MN%CD:(00BC0) Type=0.0 Nibs=2
eFSPEC =  0003A TI%ERM          - 02F04 JP%PR2(002DE) Type=0.0 Nibs=4
                                + 066FE SG%SYS(0035E) Type=0.0 Nibs=4
                                + 08294 JP%EXC(004B8) Type=0.0 Nibs=4
                                + 09F5D SG%FXQ(003E7) Type=0.0 Nibs=4
                                + 16B2D FH%TFM(0007F) Type=0.0 Nibs=2
                                + 16BC2 FH%TFM(00114) Type=0.0 Nibs=4
eFTYPE =  0003F TI%ERM          - 03E87 JP%PR3(00AB1) Type=0.0 Nibs=4
                                + 06C8A SG%SYS(008EA) Type=0.0 Nibs=4
                                + 07098 JP%SYS(00169) Type=0.0 Nibs=4
                                + 07756 JP%SYS(00827) Type=0.0 Nibs=4
                                + 0839E JP%EXC(005C2) Type=0.0 Nibs=4
```

```
                                    + 0A5CC SG%FXQ(00A56) Type=0.0 Nibs=4
                                    + 11597 SC%FIL(00544) Type=0.0 Nibs=4
                                    + 115B7 SC%FIL(00564) Type=0.0 Nibs=2
                                    + 1A231 SC%SUB(014BF) Type=0.0 Nibs=2
                                    + 1CF61 MN%CD:(006E8) Type=0.0 Nibs=2
eFnFND =  00039 TI%ERM              - 09FF8 SG%FXQ(00482) Type=0.0 Nibs=4
                                    + 11B2C SC%FIL(00AD9) Type=0.0 Nibs=4
                                    + 17235 FH%TFM(00787) Type=0.0 Nibs=4
                                    + 19622 SC%SUB(008B0) Type=0.0 Nibs=2
eFwoNX =  0002A TI%ERM              - 087E9 SG%EXC(001A1) Type=0.0 Nibs=2
eIF*ZR =  00010 TI%ERM              - 0C60B JT%MTH(002E4) Type=0.0 Nibs=2
eIF-IF =  0000F TI%ERM              - 0C689 JT%MTH(00362) Type=0.0 Nibs=2
eIF/IF =  0000E TI%ERM              - 0C65D JT%MTH(00336) Type=0.0 Nibs=2
eILCNT =  0004F TI%ERM              - 02E72 JP%PR2(0024C) Type=0.0 Nibs=4
                                    + 15A2F AB%CLC(00510) Type=0.0 Nibs=2
                                    + 19CA4 SC%SUB(00F32) Type=0.0 Nibs=2
eILEXP =  00050 TI%ERM              - 02E37 JP%PR2(00211) Type=0.0 Nibs=4
eILKEY =  00055 TI%ERM              - 1592F AB%CLC(00410) Type=0.0 Nibs=2
eILLEG =  000E6 TI%ERM              - 1F8B0 TI%ERM(004D0) Type=0.0 Nibs=2
                                    + 1FA50 TI%ERM(00670) Type=0.0 Nibs=2
eILPAR =  00051 TI%ERM              - 02E54 JP%PR2(0022E) Type=0.0 Nibs=4
                                    + 15B2C AB%CLC(0060D) Type=0.0 Nibs=2
                                    + 19A47 SC%SUB(00CD5) Type=0.0 Nibs=2
eILTFM =  00037 TI%ERM              - 16CA8 FH%TFM(001FA) Type=0.0 Nibs=4
eILVAR =  00053 TI%ERM              - 02E68 JP%PR2(00242) Type=0.0 Nibs=4
eIMGOV =  0002F TI%ERM              - 1C2D2 MB%USG(0082E) Type=0.0 Nibs=2
eINF   =  000F3 TI%ERM              - 1F447 TI%ERM(00067) Type=0.0 Nibs=2
                                    + 1F4F0 TI%ERM(00110) Type=0.0 Nibs=2
                                    + 1F4F6 TI%ERM(00116) Type=0.0 Nibs=2
                                    + 1F4FE TI%ERM(0011E) Type=0.0 Nibs=2
                                    + 1F504 TI%ERM(00124) Type=0.0 Nibs=2
                                    + 1F50C TI%ERM(0012C) Type=0.0 Nibs=2
                                    + 1F51E TI%ERM(0013E) Type=0.0 Nibs=2
                                    + 1F526 TI%ERM(00146) Type=0.0 Nibs=2
eINF^0 =  00012 TI%ERM              - 0D307 JT%MTH(00FE0) Type=0.0 Nibs=2
eINPUT =  000F4 TI%ERM              - 1F711 TI%ERM(00331) Type=0.0 Nibs=2
                                    + 1F725 TI%ERM(00345) Type=0.0 Nibs=2
                                    + 1F73A TI%ERM(0035A) Type=0.0 Nibs=2
eINVIM =  0002D TI%ERM              - 1B98D MB%IMG(00547) Type=0.0 Nibs=2
eINVLD =  000EC TI%ERM              - 1F4B9 TI%ERM(000D9) Type=0.0 Nibs=2
                                    + 1F620 TI%ERM(00240) Type=0.0 Nibs=2
                                    + 1F7A2 TI%ERM(003C2) Type=0.0 Nibs=2
                                    + 1F7B5 TI%ERM(003D5) Type=0.0 Nibs=2
                                    + 1F7DB TI%ERM(003FB) Type=0.0 Nibs=2
                                    + 1F858 TI%ERM(00478) Type=0.0 Nibs=2
                                    + 1F885 TI%ERM(004A5) Type=0.0 Nibs=2
                                    + 1F8E3 TI%ERM(00503) Type=0.0 Nibs=2
                                    + 1FA5B TI%ERM(0067B) Type=0.0 Nibs=2
                                    + 1FA6C TI%ERM(0068C) Type=0.0 Nibs=2
                                    + 1FA9C TI%ERM(006BC) Type=0.0 Nibs=2
                                    + 1FAC5 TI%ERM(006E5) Type=0.0 Nibs=2
                                    + 1FC60 TI%ERM(00880) Type=0.0 Nibs=2
eINVST =  000ED TI%ERM              - 1F80A TI%ERM(0042A) Type=0.0 Nibs=2
                                    + 1F81F TI%ERM(0043F) Type=0.0 Nibs=2
                                    + 1F832 TI%ERM(00452) Type=0.0 Nibs=2
eINVUS =  0002E TI%ERM              - 1B4EA MB%IMG(000A4) Type=0.0 Nibs=2
eINX   =  00015 TI%ERM              - 0CB71 JT%MTH(0084A) Type=0.0 Nibs=2
eIVARG =  0000B TI%ERM              - 0BF1D AB%FCN(008E4) Type=0.0 Nibs=2
                                    + 0D74D SM%MTH(00318) Type=0.0 Nibs=2
eIVSAR =  00033 TI%ERM              - 0B026 AB%REG(0042E) Type=0.0 Nibs=2
```

```
                                + 0E560 PM%STA(00731) Type=0.0 Nibs=2
eIVSOP =  00035 TI%ERM          - 0E126 PM%STA(002F7) Type=0.0 Nibs=2
                                + 0E539 PM%STA(0070A) Type=0.0 Nibs=2
eIVSTA =  00034 TI%ERM          - 0E52B PM%STA(006FC) Type=0.0 Nibs=2
eIVTAB =  00030 TI%ERM          - 17FD7 SB%IO:(0021B) Type=0.0 Nibs=4
eL2LNG =  00041 TI%ERM          - 02E04 JP%PR2(001DE) Type=0.0 Nibs=4
                                + 06C42 SG%SYS(008A2) Type=0.0 Nibs=4
                                + 1576E AB%CLC(0024F) Type=0.0 Nibs=2
                                + 15965 AB%CLC(00446) Type=0.0 Nibs=2
                                + 15B88 AB%CLC(00669) Type=0.0 Nibs=2
                                + 17763 FH%TFM(00CB5) Type=0.0 Nibs=4
                                + 17972 FH%TFM(00EC4) Type=0.0 Nibs=4
eLNO   =  0000C TI%ERM          - 0CDAE JT%MTH(00A87) Type=0.0 Nibs=2
eLOBAT =  00016 TI%ERM          - 1CFFA MN%CD:(00781) Type=0.0 Nibs=4
eLOG-  =  0000D TI%ERM          - 0CD94 JT%MTH(00A6D) Type=0.0 Nibs=2
eMEM   =  00018 TI%ERM          - 013F1 TI%UTL(002C0) Type=0.0 Nibs=4
                                + 094A0 TI%ERD(0010D) Type=0.0 Nibs=4
                                + 11722 SC%FIL(006CF) Type=0.0 Nibs=4
                                + 12327 JP%POL(0000C) Type=0.0 Nibs=4
                                + 1447B SC%DAT(00BC6) Type=0.0 Nibs=2
                                + 1CF59 MN%CD:(006E0) Type=0.0 Nibs=2
eMMCOR =  00017 TI%ERM          - 09085 SG%EXC(00A3D) Type=0.0 Nibs=2
eMPI   =  00019 TI%ERM          - 001E8 SB%DVR(001E8) Type=0.0 Nibs=2
eMSPAR =  00052 TI%ERM          - 02E5E JP%PR2(00238) Type=0.0 Nibs=4
                                + 15B12 AB%CLC(005F3) Type=0.0 Nibs=2
eNEG^X =  00009 TI%ERM          - 0D34F JT%MTH(01028) Type=0.0 Nibs=2
eNFOUN =  000E8 TI%ERM          - 1F663 TI%ERM(00283) Type=0.0 Nibs=2
                                + 1F69C TI%ERM(002BC) Type=0.0 Nibs=2
                                + 1F6BA TI%ERM(002DA) Type=0.0 Nibs=2
                                + 1F750 TI%ERM(00370) Type=0.0 Nibs=2
                                + 1F7F1 TI%ERM(00411) Type=0.0 Nibs=2
                                + 1F87D TI%ERM(0049D) Type=0.0 Nibs=2
                                + 1F906 TI%ERM(00526) Type=0.0 Nibs=2
eNODAT =  00020 TI%ERM          - 17C70 SB%RD:(000A9) Type=0.0 Nibs=2
eNOTIN =  00043 TI%ERM          - 1D541 MN%CD:(00CC8) Type=0.0 Nibs=2
eNSVAR =  00033 TI%ERM          -
eNUMIN =  00026 TI%ERM          - 18906 SB%IO:(00B4A) Type=0.0 Nibs=4
eNVSTA =  00033 TI%ERM          -
eNXMOF =  0002B TI%ERM          - 0883C SG%EXC(001F4) Type=0.0 Nibs=2
eOVFL* =  000F5 TI%ERM          - 1F652 TI%ERM(00272) Type=0.0 Nibs=2
                                + 1F6FA TI%ERM(0031A) Type=0.0 Nibs=2
                                + 1F7D3 TI%ERM(003F3) Type=0.0 Nibs=2
eOVFLW =  00002 TI%ERM          - 046F5 SB%EXP(00720) Type=0.0 Nibs=4
                                + 0CA78 JT%MTH(00751) Type=0.0 Nibs=4
                                + 161D0 AB%CLC(00CB1) Type=0.0 Nibs=2
                                + 161E4 AB%CLC(00CC5) Type=0.0 Nibs=4
ePALGN =  0005E TI%ERM          -
ePLLC  =  0005A TI%ERM          - 1CF49 MN%CD:(006D0) Type=0.0 Nibs=2
ePLLC# =  00059 TI%ERM          - 1CF69 MN%CD:(006F0) Type=0.0 Nibs=4
ePRCER =  00054 TI%ERM          - 15C9D AB%CLC(0077E) Type=0.0 Nibs=2
ePRMIS =  00024 TI%ERM          - 1596E AB%CLC(0044F) Type=0.0 Nibs=2
                                + 15CAE AB%CLC(0078F) Type=0.0 Nibs=2
                                + 15E44 AB%CLC(00925) Type=0.0 Nibs=2
                                + 167CB AB%ED:(00159) Type=0.0 Nibs=2
                                + 19478 SC%SUB(00706) Type=0.0 Nibs=2
ePRNEX =  0004C TI%ERM          - 02E97 JP%PR2(00271) Type=0.0 Nibs=4
ePROTD =  00042 TI%ERM          - 1CB3E MN%CD:(002C5) Type=0.0 Nibs=2
ePRTCT =  000F8 TI%ERM          - 1F8C8 TI%ERM(004E8) Type=0.0 Nibs=2
                                + 1F935 TI%ERM(00555) Type=0.0 Nibs=2
ePULL  =  000F6 TI%ERM          - 1F5F2 TI%ERM(00212) Type=0.0 Nibs=2
```

```
                                + 1FB23 TI%ERM(00743) Type=0.0 Nibs=2
                                + 1FB2E TI%ERM(0074E) Type=0.0 Nibs=2
eQUOEX =  0004D TI%ERM          - 02E8D JP%PR2(00267) Type=0.0 Nibs=4
eROWRN =  00056 TI%ERM          - 15AC6 AB%CLC(005A7) Type=0.0 Nibs=2
eR1WRN =  00057 TI%ERM          - 15C4A AB%CLC(00728) Type=0.0 Nibs=2
eRALGN =  0005D TI%ERM          - 1D04E MN%CD:(007D5) Type=0.0 Nibs=2
eRECOR =  0001D TI%ERM          - 1422C SC%DAT(00977) Type=0.0 Nibs=2
                                + 14523 SC%DAT(00C6E) Type=0.0 Nibs=2
eRWERR =  00046 TI%ERM          - 1CF51 MN%CD:(006D8) Type=0.0 Nibs=2
                                + 1D1A8 MN%CD:(0092F) Type=0.0 Nibs=2
eRwoGS =  0002C TI%ERM          - 09004 SG%EXC(009BC) Type=0.0 Nibs=2
eSIGOP =  00013 TI%ERM          - 0E646 PM%STA(00817) Type=0.0 Nibs=2
eSPGNF =  00031 TI%ERM          - 19128 SC%SUB(003B6) Type=0.0 Nibs=2
eSQR-  =  0000A TI%ERM          - 0C567 JT%MTH(00240) Type=0.0 Nibs=2
eSTMNF =  0001E TI%ERM          - 1B4EC MB%IMG(000A6) Type=0.0 Nibs=2
eSTROV =  00025 TI%ERM          - 0F6AD AB%ASN(000D3) Type=0.0 Nibs=2
                                + 1411C SC%DAT(00867) Type=0.0 Nibs=2
eSUBSC =  0001C TI%ERM          - 0BAA6 AB%FCN(0046D) Type=0.0 Nibs=2
eSYNTX =  0004B TI%ERM          - 02E2D JP%PR2(00207) Type=0.0 Nibs=4
                                + 02EE7 JP%PR2(002C1) Type=0.0 Nibs=4
                                + 02F69 JP%PR2(00343) Type=0.0 Nibs=4
                                + 16F79 FH%TFM(004CB) Type=0.0 Nibs=4
                                + 17739 FH%TFM(00C8B) Type=0.0 Nibs=2
eSYSER =  00017 TI%ERM          -
eTFFLD =  00038 TI%ERM          - 16EAC FH%TFM(003FE) Type=0.0 Nibs=4
eTFM   =  000F1 TI%ERM          - 1F85B TI%ERM(0047B) Type=0.0 Nibs=2
                                + 1F863 TI%ERM(00483) Type=0.0 Nibs=2
eTFWRN =  00058 TI%ERM          - 172DF FH%TFM(00831) Type=0.0 Nibs=4
eTNINF =  00004 TI%ERM          - 0D9B8 SM%MTH(00583) Type=0.0 Nibs=2
eTOO   =  000EF TI%ERM          - 1F719 TI%ERM(00339) Type=0.0 Nibs=2
                                + 1F730 TI%ERM(00350) Type=0.0 Nibs=2
                                + 1F919 TI%ERM(00539) Type=0.0 Nibs=2
                                + 1F9B0 TI%ERM(005D0) Type=0.0 Nibs=2
                                + 1F9C1 TI%ERM(005E1) Type=0.0 Nibs=2
                                + 1F9F2 TI%ERM(00612) Type=0.0 Nibs=2
eTOOFI =  00028 TI%ERM          - 188F6 SB%IO:(00B3A) Type=0.0 Nibs=4
eTOOMI =  00027 TI%ERM          - 1891B SB%IO:(00B5F) Type=0.0 Nibs=4
eTRKDN =  00061 TI%ERM          - 1CFAE MN%CD:(00735) Type=0.0 Nibs=4
eTRKOF =  000E5 TI%ERM          - 1D858 MN%CD:(00FDF) Type=0.0 Nibs=4
eTUFAS =  00047 TI%ERM          - 1D1B7 MN%CD:(0093E) Type=0.0 Nibs=2
eTUSLO =  00048 TI%ERM          - 1D1CC MN%CD:(00953) Type=0.0 Nibs=2
eUALGN =  0005F TI%ERM          - 1D739 MN%CD:(00EC0) Type=0.0 Nibs=2
eUNFLW =  00001 TI%ERM          - 046B3 SB%EXP(006DE) Type=0.0 Nibs=4
                                + 0CA5F JT%MTH(00738) Type=0.0 Nibs=4
                                + 16237 AB%CLC(00D18) Type=0.0 Nibs=2
                                + 1624E AB%CLC(00D2F) Type=0.0 Nibs=4
eUNKCD =  00045 TI%ERM          - 1CF41 MN%CD:(006C8) Type=0.0 Nibs=2
eUNORC =  00014 TI%ERM          - 0874B SG%EXC(00103) Type=0.0 Nibs=2
                                + 0D465 SM%MTH(00030) Type=0.0 Nibs=2
                                + 0E8B8 PM%STA(00A89) Type=0.0 Nibs=2
eVALGN =  0005C TI%ERM          - 1CBB9 MN%CD:(00340) Type=0.0 Nibs=2
eVARTY =  00032 TI%ERM          -
eVCNTX =  00032 TI%ERM          - 0AE2F AB%REG(00237) Type=0.0 Nibs=2
                                + 0F847 AB%ASN(0026D) Type=0.0 Nibs=2
                                + 0F8DE AB%ASN(00304) Type=0.0 Nibs=2
eVFYER =  00044 TI%ERM          - 1CBE7 MN%CD:(0036E) Type=0.0 Nibs=2
eWALGN =  0005B TI%ERM          - 1CB1D MN%CD:(002A4) Type=0.0 Nibs=2
eWRGNM =  00049 TI%ERM          - 1D308 MN%CD:(00A8F) Type=0.0 Nibs=2
eXFNNF =  00022 TI%ERM          - 0C321 AB%FCN(00CE8) Type=0.0 Nibs=2
eXWORD =  00023 TI%ERM          - 080C2 JP%EXC(002E6) Type=0.0 Nibs=2
```

```
eZRDIV =  00008 TI%ERM        - 0C4E2 JT%MTH(001BB) Type=0.0 Nibs=2
eZRO/0 =  00007 TI%ERM        - 0C4CF JT%MTH(001A8) Type=0.0 Nibs=2
enull  =  00000 TI%ERM        -
eolxc+ =  0A101 SG%FXQ        - 0642E SG%SYS(0008E) Type=1.1 Nibs=4 Dist=03CD3
                              + 06A8A SG%SYS(006EA) Type=1.1 Nibs=4 Dist=03677
eolxck =  0A104 SG%FXQ        - 0EDB1 MN%UTL(00143) Type=1.1 Nibs=4 Dist=04CAD
ew/o   =  000EB TI%ERM        - 1F75F TI%ERM(0037F) Type=0.0 Nibs=2
                              + 1F779 TI%ERM(00399) Type=0.0 Nibs=2
                              + 1F78F TI%ERM(003AF) Type=0.0 Nibs=2
expex- =  136EA PM%FLG        -
expexc =  12CF8 MN%TM:        - 1852A SB%IO:(0076E) Type=1.0 Nibs=4 Dist=05832
                              + 1A406 SB%EXC(00032) Type=1.1 Nibs=4 Dist=0770E
                              + 1A490 SB%EXC(000BC) Type=1.1 Nibs=4 Dist=07798
                              + 1A4D0 SB%EXC(000FC) Type=1.1 Nibs=4 Dist=077D8
exppar =  0387F JP%PR3        - 03284 JP%PR2(0065E) Type=1.0 Nibs=3 Dist=005FB
expr   =  1A070 SC%SUB        - 1AD8B SB%VAL(0001D) Type=1.0 Nibs=4 Dist=00D1B
                              + 1B4F7 MB%IMG(000B1) Type=1.1 Nibs=4 Dist=01487
                              + 1C435 SB%GPH(00070) Type=1.0 Nibs=4 Dist=023C5

fAOS   =  000DF TI%EQU        - 138E8 SC%DAT(00033) Type=0.0 Nibs=2
fASCII =  00001 TI%EQU        -
fBASIC =  0E214 TI%EQU        - 06ED0 SG%SYS(00B30) Type=0.0 Nibs=5
                              + 07743 JP%SYS(00814) Type=0.0 Nibs=5
                              + 0A583 SG%FXQ(00A0D) Type=0.0 Nibs=4
                              + 1180A SC%FIL(007B7) Type=0.0 Nibs=5
                              + 1706B FH%TFM(005BD) Type=0.0 Nibs=4
                              + 1A3C8 SC%SUB(01656) Type=0.0 Nibs=5
                              + 1FF56 SC%TAB(00028) Type=0.0 Nibs=4
                              + 1FF5A SC%TAB(0002C) Type=0.0 Nibs=4 Offset=      1
                              + 1FF5E SC%TAB(00030) Type=0.0 Nibs=4 Offset=      2
                              + 1FF62 SC%TAB(00034) Type=0.0 Nibs=4 Offset=      3
fBIN   =  0E204 TI%EQU        - 07083 JP%SYS(00154) Type=0.0 Nibs=5
                              + 18FDE SC%SUB(0026C) Type=0.0 Nibs=4
                              + 1A3B9 SC%SUB(01647) Type=0.0 Nibs=5
                              + 1FFCE SC%TAB(000A0) Type=0.0 Nibs=4
                              + 1FFD2 SC%TAB(000A4) Type=0.0 Nibs=4 Offset=      1
                              + 1FFD6 SC%TAB(000A8) Type=0.0 Nibs=4 Offset=      2
                              + 1FFDA SC%TAB(000AC) Type=0.0 Nibs=4 Offset=      3
fCALC? =  01C2E SB%DSP        - 0030C SB%DVR(0030C) Type=1.1 Nibs=4 Dist=01922
                              + 0033A SB%DVR(0033A) Type=1.1 Nibs=4 Dist=018F4
                              + 010B8 SB%KEY(003C1) Type=1.1 Nibs=4 Dist=00B76
                              + 094B2 TI%ERD(0011F) Type=1.1 Nibs=4 Dist=07884
                              + 097AD TI%ERD(0041A) Type=1.1 Nibs=4 Dist=07B7F
                              + 097EC TI%ERD(00459) Type=1.1 Nibs=4 Dist=07BBE
                              + 0FAB1 SC%TRC(0006A) Type=0.1 Nibs=5
                              + 19E73 SC%SUB(01101) Type=0.1 Nibs=5
fDATA  =  0E0F0 TI%EQU        - 1FF3E SC%TAB(00010) Type=0.0 Nibs=4
                              + 1FF42 SC%TAB(00014) Type=0.0 Nibs=4 Offset=      1
fEOF   =  000FF TI%EQU        - 13A5D SC%DAT(001A8) Type=0.0 Nibs=2
                              + 14202 SC%DAT(0094D) Type=0.0 Nibs=2
                              + 14332 SC%DAT(00A7D) Type=0.0 Nibs=2
                              + 14370 SC%DAT(00ABB) Type=0.0 Nibs=2
fEOR   =  000EF TI%EQU        - 13963 SC%DAT(000AE) Type=0.0 Nibs=2
                              + 1421C SC%DAT(00967) Type=0.0 Nibs=2
                              + 1433B SC%DAT(00A86) Type=0.0 Nibs=2
fEOS   =  0006F TI%EQU        - 13945 SC%DAT(00090) Type=0.0 Nibs=2
fKEY   =  0E20C TI%EQU        - 06EDF SG%SYS(00B3F) Type=0.0 Nibs=4
                              + 08392 JP%EXC(005B6) Type=0.0 Nibs=5
                              + 08BE4 SG%EXC(0059C) Type=0.0 Nibs=4
                              + 1D465 MN%CD:(00BEC) Type=0.0 Nibs=4
```

```
                                    + 1FF76 SC%TAB(00048) Type=0.0 Nibs=4
                                    + 1FF7A SC%TAB(0004C) Type=0.0 Nibs=4 Offset=      1
fLEX    =  0E208 TI%EQU             - 083E6 JP%EXC(0060A) Type=0.0 Nibs=5
                                    + 0A184 SG%FXQ(0060E) Type=0.0 Nibs=5
                                    + 10F48 MN%CNF(00DB6) Type=0.0 Nibs=5
                                    + 1FFEE SC%TAB(000C0) Type=0.0 Nibs=4
                                    + 1FFF2 SC%TAB(000C4) Type=0.0 Nibs=4 Offset=      1
                                    + 1FFF6 SC%TAB(000C8) Type=0.0 Nibs=4 Offset=      2
                                    + 1FFFA SC%TAB(000CC) Type=0.0 Nibs=4 Offset=      3
fLIF1   =  00001 TI%EQU             -
fMOS    =  0007F TI%EQU             - 1394E SC%DAT(00099) Type=0.0 Nibs=2
fSDATA  =  0E0D0 TI%EQU             -
fSOS    =  000CF TI%EQU             - 138F6 SC%DAT(00041) Type=0.0 Nibs=2
fTEXT   =  00001 TI%EQU             -
finda   =  05864 SG%LDC             - 05C76 SB%EXD(00354) Type=1.1 Nibs=3 Dist=00412
findf   =  11B32 SC%FIL             - 17364 FH%TFM(008B6) Type=1.1 Nibs=4 Dist=05832
flAC    =  FFFC7 TI%EQU             -
flALRM  =  FFFC4 TI%EQU             - 00640 SB%DVR(00640) Type=0.0 Nibs=2
flBASE  =  FFFF0 TI%EQU             - 13849 PM%FLG(00400) Type=0.0 Nibs=2
flBAT   =  FFFC3 TI%EQU             - 0EF46 MN%UTL(002D8) Type=0.0 Nibs=2
                                    + 1CFEB MN%CD:(00772) Type=0.0 Nibs=2
flBEEP  =  FFFFE TI%EQU             - 0EA84 MN%BP:(00093) Type=0.0 Nibs=2
                                    + 0EB52 MN%BP:(00161) Type=0.0 Nibs=2
flBPLD  =  FFFE7 TI%EQU             - 0EBF6 MN%BP:(00205) Type=0.0 Nibs=2
flCALC  =  FFFC0 TI%EQU             - 1556B AB%CLC(0004C) Type=0.0 Nibs=2
                                    + 16872 AB%ED:(00200) Type=0.0 Nibs=2
flCLOC  =  FFFD3 TI%EQU             - 126C6 MN%TM:(0017C) Type=0.0 Nibs=2
flCMDS  =  FFFD1 TI%EQU             - 004CA SB%DVR(004CA) Type=0.0 Nibs=2
                                    + 0163D SB%CMD(00016) Type=0.0 Nibs=2
                                    + 016C6 SB%CMD(0009F) Type=0.0 Nibs=2
flCTON  =  FFFFD TI%EQU             - 13174 MN%TM:(00C2A) Type=0.0 Nibs=2
flCTRL  =  FFFD0 TI%EQU             - 14F48 MN%ED:(002C8) Type=0.0 Nibs=2
                                    + 14F53 MN%ED:(002D3) Type=0.0 Nibs=2
                                    + 15287 MN%ED:(00607) Type=0.0 Nibs=2
flDG0   =  FFFEF TI%EQU             -
flDG1   =  FFFEE TI%EQU             -
flDG2   =  FFFED TI%EQU             -
flDG3   =  FFFEC TI%EQU             -
flDORM  =  FFFD5 TI%EQU             - 00350 SB%DVR(00350) Type=0.0 Nibs=2
                                    + 003AF SB%DVR(003AF) Type=0.0 Nibs=2
                                    + 003E1 SB%DVR(003E1) Type=0.0 Nibs=2
                                    + 0042A SB%DVR(0042A) Type=0.0 Nibs=2
flDVZ   =  FFFF9 TI%EQU             -
flEOT   =  FFFE9 TI%EQU             -
flEXAC  =  FFFD2 TI%EQU             - 128B6 MN%TM:(0036C) Type=0.0 Nibs=2
                                    + 12905 MN%TM:(003BB) Type=0.0 Nibs=2
flEXTD  =  FFFEA TI%EQU             -
flFXEN  =  FFFF3 TI%EQU             -
flINFR  =  FFFF5 TI%EQU             - 137E4 PM%FLG(0039B) Type=0.0 Nibs=2
flINX   =  FFFFC TI%EQU             -
flIVL   =  FFFF8 TI%EQU             - 1371C PM%FLG(002D3) Type=0.0 Nibs=2
flLC    =  FFFF1 TI%EQU             - 07DE8 JP%EXC(0000C) Type=0.0 Nibs=2
                                    + 1501F MN%ED:(0039F) Type=0.0 Nibs=2
                                    + 15185 MN%ED:(00505) Type=0.0 Nibs=2
                                    + 15245 MN%ED:(005C5) Type=0.0 Nibs=2
flMKOF  =  FFFCE TI%EQU             - 005AD SB%DVR(005AD) Type=0.0 Nibs=2
                                    + 006B1 SB%DVR(006B1) Type=0.0 Nibs=2
flMPI   =  FFFD7 TI%EQU             -
flNEGR  =  FFFF4 TI%EQU             - 13817 PM%FLG(003CE) Type=0.0 Nibs=2
                                    + 13824 PM%FLG(003DB) Type=0.0 Nibs=2
```

```
f1NOFN =  FFFD6 TI%EQU        - 0951B TI%ERD(00188) Type=0.0 Nibs=2
                              + 18E04 SC%SUB(00092) Type=0.0 Nibs=2
                              + 19150 SC%SUB(003DE) Type=0.0 Nibs=2
                              + 19A3A SC%SUB(00CC8) Type=0.0 Nibs=2
f1NOPR =  FFFE6 TI%EQU        - 00358 SB%DVR(00358) Type=0.0 Nibs=2
                              + 00394 SB%DVR(00394) Type=0.0 Nibs=2
f1NZ4  =  FFFE8 TI%EQU        -
f1OVF  =  FFFFA TI%EQU        - 161DA AB%CLC(00CBB) Type=0.0 Nibs=2
f1PDWN =  FFFEB TI%EQU        -
f1PRGM =  FFFC2 TI%EQU        - 07358 JP%SYS(00429) Type=0.0 Nibs=2
                              + 076E0 JP%SYS(007B1) Type=0.0 Nibs=2
f1PWDN =  FFFCF TI%EQU        - 0052C SB%DVR(0052C) Type=0.0 Nibs=2
                              + 00573 SB%DVR(00573) Type=0.0 Nibs=2
f1QIET =  FFFFF TI%EQU        - 093D7 TI%ERD(00044) Type=0.0 Nibs=2
f1RAD  =  FFFF6 TI%EQU        - 13865 PM%FLG(0041C) Type=0.0 Nibs=2
                              + 13877 PM%FLG(0042E) Type=0.0 Nibs=2
f1RPTD =  FFFC5 TI%EQU        - 152BC MN%ED:(0063C) Type=0.0 Nibs=2
                              + 152D7 MN%ED:(00657) Type=0.0 Nibs=2
                              + 1532E MN%ED:(006AE) Type=0.0 Nibs=2
f1RTN  =  FFFD4 TI%EQU        - 0260F JP%PR1(0006F) Type=0.0 Nibs=2
                              + 02BFA JP%PR1(0065A) Type=0.0 Nibs=2
f1SCEN =  FFFF2 TI%EQU        -
f1SUSP =  FFFC1 TI%EQU        - 07382 JP%SYS(00453) Type=0.0 Nibs=2
                              + 07993 JP%SYS(00A64) Type=0.0 Nibs=2
                              + 07B78 JP%SYS(00C49) Type=0.0 Nibs=2
f1TNOF =  FFFCD TI%EQU        - 005A5 SB%DVR(005A5) Type=0.0 Nibs=2
                              + 00616 SB%DVR(00616) Type=0.0 Nibs=2
                              + 00626 SB%DVR(00626) Type=0.0 Nibs=2
                              + 006BC SB%DVR(006BC) Type=0.0 Nibs=2
                              + 18C9E MN%LCK(00046) Type=0.0 Nibs=2
                              + 18CBB MN%LCK(00063) Type=0.0 Nibs=2
f1UNF  =  FFFFB TI%EQU        - 16244 AB%CLC(00D25) Type=0.0 Nibs=2
f1USER =  FFFF7 TI%EQU        - 010A8 SB%KEY(003B1) Type=0.0 Nibs=2
                              + 07DFA JP%EXC(0001E) Type=0.0 Nibs=2
                              + 14F88 MN%ED:(00308) Type=0.0 Nibs=2
                              + 14F94 MN%ED:(00314) Type=0.0 Nibs=2
                              + 1524F MN%ED:(005CF) Type=0.0 Nibs=2
                              + 1528F MN%ED:(0060F) Type=0.0 Nibs=2
f1USRX =  FFFC6 TI%EQU        - 14F75 MN%ED:(002F5) Type=0.0 Nibs=2
                              + 14F80 MN%ED:(00300) Type=0.0 Nibs=2
                              + 15297 MN%ED:(00617) Type=0.0 Nibs=2
f1VIEW =  FFFCC TI%EQU        - 14EB3 MN%ED:(00233) Type=0.0 Nibs=2
                              + 14EC2 MN%ED:(00242) Type=0.0 Nibs=2
                              + 152A1 MN%ED:(00621) Type=0.0 Nibs=2
f1tdh  =  0BDAA AB%FCN        - 066B0 SG%SYS(00310) Type=1.1 Nibs=4 Dist=056FA
                              + 080AE JP%EXC(002D2) Type=1.1 Nibs=4 Dist=03CFC
                              + 09F0E SG%FXQ(00398) Type=1.1 Nibs=4 Dist=01E9C
                              + 0AD78 AB%REG(00180) Type=1.1 Nibs=4 Dist=01032
                              + 0EFE9 MN%UTL(0037B) Type=1.0 Nibs=4 Dist=0323F
                              + 1119B SC%FIL(00148) Type=1.1 Nibs=4 Dist=053F1
fpoll  =  13ABF SC%DAT        - 1B999 MB%IMG(00553) Type=1.0 Nibs=4 Dist=07EDA
fspecx =  117E0 SC%FIL        - 16B09 FH%TFM(0005B) Type=1.1 Nibs=4 Dist=05329
                              + 190EF SC%SUB(0037D) Type=1.1 Nibs=4 Dist=0790F

hdflt  =  0BEC7 AB%FCN        - 0A9E2 SB%FCN(000B7) Type=1.1 Nibs=4 Dist=014E5
                              + 0E669 PM%STA(0083A) Type=1.1 Nibs=4 Dist=027A2

i/odal =  1CF0A MN%CD:        - 1A555 SB%EXC(00181) Type=1.1 Nibs=4 Dist=029B5
i/ofnd =  097A4 TI%ERD        - 08FC2 SG%EXC(0097A) Type=1.0 Nibs=3 Dist=007E2
inexit =  188EE SB%IO:        - 18C95 MN%LCK(0003D) Type=1.0 Nibs=3 Dist=003A7
```

```
initpt =  01441 TI%UTL        - 017D4 SB%CMD(001AD) Type=1.1 Nibs=3 Dist=00393
                              + 07D61 JP%SYS(00E32) Type=1.1 Nibs=4 Dist=06920

k#-CHR =  00068 SB%KCM        -
k#-LIN =  0006B SB%KCM        -
k#1    =  00027 SB%KCM        - 01074 SB%KEY(0037D) Type=0.0 Nibs=2
k#2    =  00028 SB%KCM        - 01079 SB%KEY(00382) Type=0.0 Nibs=2
k#3    =  00029 SB%KCM        - 0107E SB%KEY(00387) Type=0.0 Nibs=2
k#ATTN =  0002B SB%KCM        - 1D072 MN%CD:(007F9) Type=0.0 Nibs=2
k#BKSP =  00067 SB%KCM        -
k#BOT  =  000A3 SB%KCM        - 064F9 SG%SYS(00159) Type=0.0 Nibs=2
k#CALC =  0006F SB%KCM        -
k#CONT =  00070 SB%KCM        -
k#CTRL =  0009E SB%KCM        -
k#DOWN =  00033 SB%KCM        - 064EA SG%SYS(0014A) Type=0.0 Nibs=2
k#EOL  =  00026 SB%KCM        - 01083 SB%KEY(0038C) Type=0.0 Nibs=2
                              + 1D06D MN%CD:(007F4) Type=0.0 Nibs=2
k#FLFT =  0009F SB%KCM        - 0219A SB%DSP(0090A) Type=0.0 Nibs=2
k#FRT  =  000A0 SB%KCM        - 021A4 SB%DSP(00914) Type=0.0 Nibs=2
k#GON  =  0009B SB%KCM        -
k#I/R  =  00069 SB%KCM        -
k#LAST =  000A4 SB%KCM        -
k#LC   =  0006A SB%KCM        -
k#LERR =  000A1 SB%KCM        -
k#LFT  =  0002F SB%KCM        - 02195 SB%DSP(00905) Type=0.0 Nibs=2
k#OFF  =  00063 SB%KCM        - 14E66 MN%ED:(001E6) Type=0.0 Nibs=2
                              + 1D077 MN%CD:(007FE) Type=0.0 Nibs=2
k#RT   =  00030 SB%KCM        - 0219F SB%DSP(0090F) Type=0.0 Nibs=2
k#RUN  =  0002E SB%KCM        -
k#SST  =  00066 SB%KCM        -
k#TOP  =  000A2 SB%KCM        - 064F4 SG%SYS(00154) Type=0.0 Nibs=2
k#UP   =  00032 SB%KCM        - 064EF SG%SYS(0014F) Type=0.0 Nibs=2
k#USER =  0006D SB%KCM        -
k#USEX =  000A5 SB%KCM        -
k#VIEW =  0006E SB%KCM        -
kMFERR =  1B4EE MB%IMG        - 1C2F3 MB%USG(0084F) Type=1.0 Nibs=4 Dist=00E05
kc-CHR =  00000 SB%KCM        -
kc-LIN =  00004 SB%KCM        -
kcATTN =  0000E SB%KCM        - 1872B SB%IO:(0096F) Type=0.0 Nibs=2
kcBKSP =  00007 SB%KCM        -
kcBOT  =  00015 SB%KCM        - 15776 AB%CLC(00257) Type=0.0 Nibs=2
                              + 18744 SB%IO:(00988) Type=0.0 Nibs=2
kcCALC =  00017 SB%KCM        -
kcCONT =  00010 SB%KCM        - 18730 SB%IO:(00974) Type=0.0 Nibs=2
kcCTRL =  0000A SB%KCM        - 14FFC MN%ED:(0037C) Type=0.0 Nibs=2
kcDOWN =  00013 SB%KCM        - 1873A SB%IO:(0097E) Type=0.0 Nibs=2
kcEOL  =  0000D SB%KCM        -
kcFLFT =  00005 SB%KCM        -
kcFRT  =  00006 SB%KCM        -
kcGON  =  00016 SB%KCM        -
kcI/R  =  00002 SB%KCM        -
kcLAST =  00019 SB%KCM        - 18749 SB%IO:(0098D) Type=0.0 Nibs=2
kcLC   =  00001 SB%KCM        - 14FF2 MN%ED:(00372) Type=0.0 Nibs=2
kcLERR =  0001A SB%KCM        - 1500B MN%ED:(0038B) Type=0.0 Nibs=2
kcLFT  =  00008 SB%KCM        -
kcOFF  =  00018 SB%KCM        - 1589B AB%CLC(0037C) Type=0.0 Nibs=5
                              + 1874E SB%IO:(00992) Type=0.0 Nibs=2
kcRT   =  00009 SB%KCM        -
kcRUN  =  0000F SB%KCM        -
kcSST  =  00011 SB%KCM        - 16964 AB%ED:(002F2) Type=0.0 Nibs=2
```

```
kcTOP  =  00014 SB%KCM        - 1873F SB%IO:(00983) Type=0.0 Nibs=2
kcUP   =  00012 SB%KCM        - 18735 SB%IO:(00979) Type=0.0 Nibs=2
kcUSER =  00003 SB%KCM        - 14FF7 MN%ED:(00377) Type=0.0 Nibs=2
kcUSEX =  0000C SB%KCM        - 15006 MN%ED:(00386) Type=0.0 Nibs=2
kcVIEW =  0000B SB%KCM        - 15001 MN%ED:(00381) Type=0.0 Nibs=2
keyfnd =  15191 MN%ED:        - 1AAEC SG%KEY(0004B) Type=1.1 Nibs=4 Dist=0595B
keyscn =  00155 SB%DVR        - 001D3 SB%DVR(001D3) Type=1.1 Nibs=3 Dist=0007E

lACCSb =  00001 TI%EQU        -
lAp    =  00010 TI%EQU        -
lBPOSp =  00005 TI%EQU        -
lCOPYb =  00001 TI%EQU        -
lCPOSb =  00006 TI%EQU        -
lDOp   =  00005 TI%EQU        -
lD1p   =  00005 TI%EQU        -
lDATEh =  00006 TI%EQU        -
lDBEGb =  0000B TI%EQU        - 1720D FH%TFM(0075F) Type=0.0 Nibs=1 Offset=     -1
lDEVC  =  00005 TI%EQU        -
lDEVCb =  00001 TI%EQU        -
lDLENb =  00006 TI%EQU        -
lDp    =  00010 TI%EQU        -
lEOL   =  00002 TI%EQU        -
lFBEGb =  00006 TI%EQU        - 171A9 FH%TFM(006FB) Type=0.0 Nibs=1 Offset=     -1
lFBF#b =  00003 TI%EQU        -
lFIB   =  0003F TI%EQU        - 11BA7 SC%FIL(00B54) Type=0.0 Nibs=2
                              + 11BE8 SC%FIL(00B95) Type=0.0 Nibs=2
                              + 11C29 SC%FIL(00BD6) Type=0.0 Nibs=2
                              + 120D5 SC%FIL(01082) Type=0.0 Nibs=2
                              + 121FF SC%FIL(011AC) Type=0.0 Nibs=2
                              + 12300 SC%FIL(012AD) Type=0.0 Nibs=2
lFIL#b =  00002 TI%EQU        -
lFILSV =  00032 TI%EQU        - 01533 TI%UTL(00402) Type=0.0 Nibs=5
                              + 015E7 TI%UTL(004B6) Type=0.0 Nibs=5
lFLAGh =  00002 TI%EQU        -
lFLENh =  00005 TI%EQU        - 117F5 SC%FIL(007A2) Type=0.0 Nibs=1 Offset=     -1
lFNAM+ =  00004 TI%EQU        -
lFNAM8 =  00010 TI%EQU        -
lFNAMh =  00010 TI%EQU        -
lFSIZb =  00006 TI%EQU        -
lFTYPb =  00004 TI%EQU        -
lFTYPh =  00004 TI%EQU        -
lLXADR =  00005 TI%EQU        - 099DC TI%ERD(00649) Type=0.0 Nibs=1 Offset=     -1
                              + 09A0C TI%ERD(00679) Type=0.0 Nibs=1 Offset=     -1
lLXENT =  0000B TI%EQU        - 099D4 TI%ERD(00641) Type=0.0 Nibs=1 Offset=     -1
                              + 10FA9 MN%CNF(00E17) Type=0.0 Nibs=1
lLXFAD =  00005 TI%EQU        -
lLXID  =  00002 TI%EQU        -
lLXTKR =  00004 TI%EQU        -
lMSGp  =  00004 TI%EQU        -
lPOL#p =  00005 TI%EQU        -
lPOLLp =  00005 TI%EQU        -
lPOLSV =  0003E TI%EQU        - 12333 JP%POL(00018) Type=0.0 Nibs=2 Offset=      6
lPOLra =  00006 TI%EQU        -
lPROTb =  00001 TI%EQU        -
lREC#b =  00004 TI%EQU        -
lRECLb =  00004 TI%EQU        -
lRLENb =  00005 TI%EQU        -
lRTN1p =  00005 TI%EQU        -
lRTN2p =  00005 TI%EQU        -
lRTN3p =  00005 TI%EQU        -
```

```
lSHLNb =  00002 TI%EQU      -
lSPDTB =  0004E TI%EQU      -
lSPDn  =  00001 TI%EQU      -
lSPDn2 =  00001 TI%EQU      -
lTEXTp =  00004 TI%EQU      -
lTIMEh =  00004 TI%EQU      -
lsleep =  00190 SB%DVR      -

makebf =  0015C SB%DVR      -
mferrs =  16B70 FH%TFM      - 18B5D SB%IO:(00DA1) Type=1.0 Nibs=4 Dist=01FED

nokeys =  0247D SB%DSP      - 004E5 SB%DVR(004E5) Type=1.1 Nibs=4 Dist=01F98
                            + 00638 SB%DVR(00638) Type=1.1 Nibs=4 Dist=01E45
                            + 0100B SB%KEY(00314) Type=1.1 Nibs=4 Dist=01472
nomem  =  118F3 SC%FIL      - 17F82 SB%IO:(001C6) Type=1.0 Nibs=4 Dist=0638F
ntken+ =  02BC4 JP%PR1      - 02EA3 JP%PR2(0027D) Type=1.1 Nibs=3 Dist=002DF

o41sod =  00005 TI%EQU      - 1FFAD SC%TAB(0007F) Type=0.0 Nibs=2
oACCSb =  0000B TI%EQU      -
oAp    =  0003E TI%EQU      -
oBNsod =  00011 TI%EQU      - 1FFC1 SC%TAB(00093) Type=0.0 Nibs=2
oBPOSp =  00005 TI%EQU      -
oBSsod =  00011 TI%EQU      - 1FF49 SC%TAB(0001B) Type=0.0 Nibs=2
oCOPYb =  0000A TI%EQU      -
oCPOSb =  00028 TI%EQU      - 13B3A SC%DAT(00285) Type=0.0 Nibs=2
oDOp   =  00019 TI%EQU      -
oD1p   =  0001E TI%EQU      -
oDATEh =  0001A TI%EQU      -
oDAsod =  0000D TI%EQU      - 1FF31 SC%TAB(00003) Type=0.0 Nibs=2
oDBEGb =  00015 TI%EQU      - 11237 SC%FIL(001E4) Type=0.0 Nibs=2
oDEVCb =  0000C TI%EQU      -
oDLENb =  0002E TI%EQU      -
oDp    =  0002E TI%EQU      -
oFBEGb =  0000D TI%EQU      - 171A5 FH%TFM(006F7) Type=0.0 Nibs=1 Offset=     -1
oFBF#b =  00002 TI%EQU      -
oFIL#b =  00000 TI%EQU      -
oFLAGh =  00014 TI%EQU      - 085EA JP%EXC(0080E) Type=0.0 Nibs=2
                            + 11DD4 SC%FIL(00D81) Type=0.0 Nibs=2
oFLENh =  00020 TI%EQU      - 06517 SG%SYS(00177) Type=0.0 Nibs=2
                            + 06F21 SG%SYS(00B81) Type=0.0 Nibs=2
                            + 07C20 JP%SYS(00CF1) Type=0.0 Nibs=2
                            + 083AC JP%EXC(005D0) Type=0.0 Nibs=2
                            + 085BF JP%EXC(007E3) Type=0.0 Nibs=2
                            + 11784 SC%FIL(00731) Type=0.0 Nibs=2
                            + 11C85 SC%FIL(00C32) Type=0.0 Nibs=2
                            + 11EF0 SC%FIL(00E9D) Type=0.0 Nibs=2
                            + 1C7BD SG%POK(002CF) Type=0.0 Nibs=2
oFLSTr =  00031 TI%EQU      - 0A224 SG%FXQ(006AE) Type=0.0 Nibs=2
                            + 17A78 MN%FRP(00045) Type=0.0 Nibs=2
                            + 1A9A3 SC%REN(00427) Type=0.0 Nibs=2
oFNAMh =  00000 TI%EQU      -
oFSIZb =  00039 TI%EQU      -
oFT-FL =  00010 TI%EQU      -
oFTYPb =  00005 TI%EQU      -
oFTYPh =  00010 TI%EQU      - 16F40 FH%TFM(00492) Type=0.0 Nibs=1 Offset=     -1
oIMPLh =  00025 TI%EQU      - 11CAE SC%FIL(00C5B) Type=0.0 Nibs=2
                            + 11FF3 SC%FIL(00FA0) Type=0.0 Nibs=2 Offset=      8
oKYsod =  00005 TI%EQU      - 08CE1 SG%EXC(00699) Type=0.0 Nibs=2
                            + 1FF69 SC%TAB(0003B) Type=0.0 Nibs=2
oLXsod =  00005 TI%EQU      - 1FFE1 SC%TAB(000B3) Type=0.0 Nibs=2
```

```
oMAINT =  0005D TI%EQU       - 10F79 MN%CNF(00DE7) Type=0.0 Nibs=2
oMSGPT =  00009 TI%EQU       - 099E5 TI%ERD(00652) Type=0.0 Nibs=1 Offset=     -1
oPOL#p =  0000A TI%EQU       -
oPROTb =  00009 TI%EQU       - 171C0 FH%TFM(00712) Type=0.0 Nibs=1 Offset=     -1
oREC#b =  00020 TI%EQU       -
oRECLb =  00024 TI%EQU       -
oRLENb =  00034 TI%EQU       -
oRTN1p =  0000A TI%EQU       -
oRTN2p =  0000F TI%EQU       -
oRTN3p =  00014 TI%EQU       -
oSHLNb =  00013 TI%EQU       -
oSPDTB =  00111 TI%EQU       - 04A71 AB%LEX(0018B) Type=0.0 Nibs=3
oSPDn2 =  0000E TI%EQU       - 04AA0 AB%LEX(001BA) Type=0.0 Nibs=1 Offset=     -1
                             + 050B9 SG%LDC(0015B) Type=0.0 Nibs=1
oSUBLn =  00025 TI%EQU       - 1A2FE SC%SUB(0158C) Type=0.0 Nibs=5
                             + 1A73A SC%REN(001BE) Type=0.0 Nibs=2
oTIMEh =  00016 TI%EQU       -
oTXsod =  00005 TI%EQU       - 1FF81 SC%TAB(00053) Type=0.0 Nibs=2
                             + 1FF99 SC%TAB(0006B) Type=0.0 Nibs=2
out2tc =  0581A SG%LDC       - 05A9D SB%EXD(0017B) Type=1.1 Nibs=3 Dist=00283
                             + 05F5A SB%EXD(00638) Type=1.1 Nibs=3 Dist=00740
outbyt =  05CC6 SB%EXD       - 056E1 SG%LDC(00783) Type=1.0 Nibs=3 Dist=005E5
                             + 056E9 SG%LDC(0078B) Type=1.0 Nibs=3 Dist=005DD
                             + 056F1 SG%LDC(00793) Type=1.0 Nibs=3 Dist=005D5
                             + 056FC SG%LDC(0079E) Type=1.1 Nibs=3 Dist=005CA
                             + 0590A SG%LDC(009AC) Type=1.1 Nibs=3 Dist=003BC

pBSCen =  000F5 TI%EQU       - 07444 JP%SYS(00515) Type=0.0 Nibs=2
pBSCex =  000F6 TI%EQU       - 07617 JP%SYS(006E8) Type=0.0 Nibs=2
pCALRS =  00036 TI%EQU       - 19701 SC%SUB(0098F) Type=0.0 Nibs=2
pCALSV =  00037 TI%EQU       - 19086 SC%SUB(00314) Type=0.0 Nibs=2
pCAT   =  00006 TI%EQU       - 065CF SG%SYS(0022F) Type=0.0 Nibs=2
pCAT$  =  00007 TI%EQU       - 066F2 SG%SYS(00352) Type=0.0 Nibs=2
pCLDST =  000FF TI%EQU       - 002FB SB%DVR(002FB) Type=0.0 Nibs=2
pCMPLX =  00038 TI%EQU       - 0BC60 AB%FCN(00627) Type=0.0 Nibs=2
pCONFG =  000FB TI%EQU       - 10F0A MN%CNF(00D78) Type=0.0 Nibs=2
pCOPYx =  00008 TI%EQU       - 08278 JP%EXC(0049C) Type=0.0 Nibs=2
pCRDAB =  00033 TI%EQU       - 1D4F7 MN%CD:(00C7E) Type=0.0 Nibs=2
pCREAT =  00009 TI%EQU       - 1174B SC%FIL(006F8) Type=0.0 Nibs=2
pCRT=8 =  00023 TI%EQU       - 116DE SC%FIL(0068B) Type=0.0 Nibs=2
pCURSR =  00029 TI%EQU       - 10076 JP%MEM(00117) Type=0.0 Nibs=2
pDATLN =  0002A TI%EQU       - 11E6F SC%FIL(00E1C) Type=0.0 Nibs=2
pDEVCp =  00001 TI%EQU       - 03D5B JP%PR3(00985) Type=0.0 Nibs=2
pDIDST =  0000A TI%EQU       - 11E0B SC%FIL(00DB8) Type=0.0 Nibs=2
pDSWKY =  000FD TI%EQU       - 0064A SB%DVR(0064A) Type=0.0 Nibs=2
pDSWNK =  000FE TI%EQU       - 00620 SB%DVR(00620) Type=0.0 Nibs=2
pEDIT  =  0002B TI%EQU       - 0A5C0 SG%FXQ(00A4A) Type=0.0 Nibs=2
pENTER =  00012 TI%EQU       -
pEOFIL =  00025 TI%EQU       - 13FEE SC%DAT(00739) Type=0.0 Nibs=2
pERROR =  000F2 TI%EQU       - 093F5 TI%ERD(00062) Type=0.0 Nibs=2
pExcpt =  000F8 TI%EQU       - 0752F JP%SYS(00600) Type=0.0 Nibs=2
pFASCH =  0002C TI%EQU       - 110EA SC%FIL(00097) Type=0.0 Nibs=2
pFILDC =  00002 TI%EQU       - 057C9 SG%LDC(0086B) Type=0.0 Nibs=2
pFILXQ =  00003 TI%EQU       - 09CBE SG%FXQ(00148) Type=0.0 Nibs=2
pFINDF =  00017 TI%EQU       - 11B1C SC%FIL(00AC9) Type=0.0 Nibs=2
                             + 11FAA SC%FIL(00F57) Type=0.0 Nibs=2
pFNIN  =  0003D TI%EQU       - 19DDA SC%SUB(01068) Type=0.0 Nibs=2
pFNOUT =  0003E TI%EQU       - 19EC0 SC%SUB(0114E) Type=0.0 Nibs=2
pFPROT =  0000B TI%EQU       - 0A430 SG%FXQ(008BA) Type=0.0 Nibs=2
pFSPCp =  00004 TI%EQU       - 03D23 JP%PR3(0094D) Type=0.0 Nibs=2
```

```
pFSPCx =  00005 TI%EQU        - 09F51 SG%FXQ(003DB) Type=0.0 Nibs=2
pFTYPE =  0002D TI%EQU        - 11071 SC%FIL(0001E) Type=0.0 Nibs=2
pIMCHR =  0001E TI%EQU        - 1B57F MB%IMG(00139) Type=0.0 Nibs=2
pIMXCH =  0001F TI%EQU        - 1BB93 MB%USG(000EF) Type=0.0 Nibs=2
pIMXQT =  0001D TI%EQU        - 1BADA MB%USG(00036) Type=0.0 Nibs=2
pIMbck =  00020 TI%EQU        - 1B984 MB%IMG(0053E) Type=0.0 Nibs=2
pIMcpi =  00021 TI%EQU        - 1B8BD MB%IMG(00477) Type=0.0 Nibs=2
pIMcpw =  00022 TI%EQU        - 1BD05 MB%USG(00261) Type=0.0 Nibs=2
pKYDF  =  0001B TI%EQU        - 14EAA MN%ED:(0022A) Type=0.0 Nibs=2
pLIST  =  0000C TI%EQU        - 06A51 SG%SYS(006B1) Type=0.0 Nibs=2
pLIST2 =  0002E TI%EQU        - 06A5F SG%SYS(006BF) Type=0.0 Nibs=2
pMEM   =  000F1 TI%EQU        - 09471 TI%ERD(000DE) Type=0.0 Nibs=2
pMERGE =  0000D TI%EQU        - 06C70 SG%SYS(008D0) Type=0.0 Nibs=2
pMNLP  =  000FA TI%EQU        - 00317 SB%DVR(00317) Type=0.0 Nibs=2
pMRGE2 =  0002F TI%EQU        - 06C7E SG%SYS(008DE) Type=0.0 Nibs=2
pPARSE =  000F4 TI%EQU        - 0263A JP%PR1(0009A) Type=0.0 Nibs=2
pPOLL2 =  0A2F8 SG%FXQ        - 065D3 SG%SYS(00233) Type=1.0 Nibs=4 Dist=03D25
pPOLL3 =  06C80 SG%SYS        -
pPRGPR =  00032 TI%EQU        - 0A10E SG%FXQ(00598) Type=0.0 Nibs=2
pPRIN# =  00026 TI%EQU        - 142AB SC%DAT(009F6) Type=0.0 Nibs=2
pPRTCL =  0000E TI%EQU        - 185CC SB%IO:(00810) Type=0.0 Nibs=2
pPRTIS =  0000F TI%EQU        - 1859D SB%IO:(007E1) Type=0.0 Nibs=2
pPURGE =  00010 TI%EQU        - 0A0A3 SG%FXQ(0052D) Type=0.0 Nibs=2
                              + 1737A FH%TFM(008CC) Type=0.0 Nibs=2
pPWROF =  000FC TI%EQU        - 005A1 SB%DVR(005A1) Type=0.0 Nibs=2
pRCRD  =  00034 TI%EQU        - 1D4C9 MN%CD:(00C50) Type=0.0 Nibs=2
pRDCBF =  00018 TI%EQU        - 1141F SC%FIL(003CC) Type=0.0 Nibs=2
                              + 13B63 SC%DAT(002AE) Type=0.0 Nibs=2
pRDNBF =  00019 TI%EQU        - 13A55 SC%DAT(001A0) Type=0.0 Nibs=2
                              + 13AA4 SC%DAT(001EF) Type=0.0 Nibs=2
pREAD# =  00027 TI%EQU        - 13ED5 SC%DAT(00620) Type=0.0 Nibs=2
pREN   =  00039 TI%EQU        - 1A87A SC%REN(002FE) Type=0.0 Nibs=2
pRNAME =  00011 TI%EQU        - 0A2F6 SG%FXQ(00780) Type=0.0 Nibs=2
pRTNTp =  0003A TI%EQU        - 09014 SG%EXC(009CC) Type=0.0 Nibs=2
pRUNft =  00030 TI%EQU        - 07091 JP%SYS(00162) Type=0.0 Nibs=2
pRUNnB =  00031 TI%EQU        - 071E5 JP%SYS(002B6) Type=0.0 Nibs=2
pSREC# =  00028 TI%EQU        - 11226 SC%FIL(001D3) Type=0.0 Nibs=2
pSREQ  =  000F9 TI%EQU        - 0076C SB%DVR(0076C) Type=0.0 Nibs=2
pTEST  =  000F0 TI%EQU        -
pTIMR# =  0003B TI%EQU        - 080E1 JP%EXC(00305) Type=0.0 Nibs=2
pTRANS =  000EF TI%EQU        -
pTRFMx =  0003C TI%EQU        - 17046 FH%TFM(00598) Type=0.0 Nibs=2
pVER$  =  00000 TI%EQU        - 0A994 SB%FCN(00069) Type=0.0 Nibs=2
pWARN  =  000F3 TI%EQU        - 093CB TI%ERD(00038) Type=0.0 Nibs=2
pWCRD  =  00035 TI%EQU        - 1CB19 MN%CD:(002A0) Type=0.0 Nibs=2
pWCRD8 =  00024 TI%EQU        - 1C9A7 MN%CD:(0012E) Type=0.0 Nibs=2
pWRCBF =  0001A TI%EQU        - 113FF SC%FIL(003AC) Type=0.0 Nibs=2
                              + 12230 SC%FIL(011DD) Type=0.0 Nibs=2
pWTKY  =  0001C TI%EQU        - 14E1E MN%ED:(0019E) Type=0.0 Nibs=2
pZERPG =  000F7 TI%EQU        - 07D8D JP%SYS(00E5E) Type=0.0 Nibs=2
polld+ =  03D8D JP%PR3        - 057C5 SG%LDC(00867) Type=1.1 Nibs=4 Dist=01A38
popbuf =  00163 SB%DVR        -
popmth =  1852E SB%IO:        - 17D70 SB%RD:(001A9) Type=1.1 Nibs=3 Dist=007BE

r<rst2 =  1CF31 MN%CD:        - 18CC7 MN%LCK(0006F) Type=1.1 Nibs=4 Dist=0426A
r<rstk =  17294 FH%TFM        - 1ADBC SB%VAL(0004E) Type=1.1 Nibs=4 Dist=03B28
reconf =  0017F SB%DVR        -
romchk =  0A447 SG%FXQ        - 065F1 SG%SYS(00251) Type=1.1 Nibs=4 Dist=03E56
                              + 06645 SG%SYS(002A5) Type=1.1 Nibs=4 Dist=03E02
                              + 08551 JP%EXC(00775) Type=1.1 Nibs=4 Dist=01EF6
```

```
romfnd =  0A44D SG%FXQ        - 06628 SG%SYS(00288) Type=1.1 Nibs=4 Dist=03E25
                              + 08561 JP%EXC(00785) Type=1.1 Nibs=4 Dist=01EEC
rptky  =  0016A SB%DVR        -
rst2<r =  1CF38 MN%CD:        - 18CFA MN%LCK(000A2) Type=1.1 Nibs=4 Dist=0423E
                              + 18D50 MN%LCK(000F8) Type=1.1 Nibs=4 Dist=041E8
rstd1  =  08C44 SG%EXC        - 0BFC8 AB%FCN(0098F) Type=1.1 Nibs=4 Dist=03384
rstk<r =  1728C FH%TFM        - 1AE09 SB%VAL(0009B) Type=1.1 Nibs=4 Dist=03B7D
rstlvl =  1CF38 MN%CD:        - 150F0 MN%ED:(00470) Type=1.1 Nibs=4 Dist=07E48
rstst  =  16320 AB%CLC        - 19000 SC%SUB(0028E) Type=1.1 Nibs=4 Dist=02CE0
rtncc  =  1364A PM%FLG        -

sARITH =  00007 TI%EQU        -
sBYEx  =  00000 TI%EQU        -
sC/P   =  00001 MB%IMG        -
sCARD  =  00002 TI%EQU        - 011F9 TI%UTL(000C8) Type=0.0 Nibs=1
                              + 012E7 TI%UTL(001B6) Type=0.0 Nibs=1
                              + 01305 TI%UTL(001D4) Type=0.0 Nibs=1
                              + 16BB6 FH%TFM(00108) Type=0.0 Nibs=1
sCARDC =  00008 TI%EQU        -
sCHAIN =  0000B TI%EQU        -
sCONT  =  0000A TI%EQU        -
sCONTK =  00009 TI%EQU        -
sCURBT =  00003 TI%EQU        -
sCURUD =  00004 TI%EQU        -
sCURUP =  00002 TI%EQU        -
sCntg  =  00002 MB%IMG        -
sCplxP =  00007 MB%IMG        -
sDEST  =  00003 TI%EQU        - 01240 TI%UTL(0010F) Type=0.0 Nibs=1
                              + 16AC4 FH%TFM(00016) Type=0.0 Nibs=1
                              + 16ADE FH%TFM(00030) Type=0.0 Nibs=1
                              + 16B38 FH%TFM(0008A) Type=0.0 Nibs=1
                              + 16B3D FH%TFM(0008F) Type=0.0 Nibs=1
                              + 16DF7 FH%TFM(00349) Type=0.0 Nibs=1
                              + 16FBD FH%TFM(0050F) Type=0.0 Nibs=1
                              + 1727B FH%TFM(007CD) Type=0.0 Nibs=1
                              + 17282 FH%TFM(007D4) Type=0.0 Nibs=1
sENDx  =  00001 TI%EQU        -
sEOF   =  00007 TI%EQU        - 17021 FH%TFM(00573) Type=0.0 Nibs=1
                              + 1710B FH%TFM(0065D) Type=0.0 Nibs=1
                              + 17113 FH%TFM(00665) Type=0.0 Nibs=1
                              + 1749D FH%TFM(009EF) Type=0.0 Nibs=1
                              + 1788B FH%TFM(00DDD) Type=0.0 Nibs=1
sERROR =  00000 TI%EQU        -
sEXTDV =  00000 TI%EQU        - 011D0 TI%UTL(0009F) Type=0.0 Nibs=1
                              + 012FA TI%UTL(001C9) Type=0.0 Nibs=1
sEXTGS =  00005 TI%EQU        -
sFOUND =  0000A MB%IMG        -
sGOSUB =  00003 TI%EQU        -
sI/OBF =  0000A FH%TFM        - 17845 FH%TFM(00D97) Type=0.0 Nibs=1
                              + 179AB FH%TFM(00EFD) Type=0.0 Nibs=1
                              + 179C2 FH%TFM(00F14) Type=0.0 Nibs=1
sINFRD =  0000A JT%MTH        - 0F9DA AB%ASN(00400) Type=0.0 Nibs=1
sINX   =  00005 JT%MTH        - 0D650 SM%MTH(0021B) Type=0.0 Nibs=1
                              + 0D658 SM%MTH(00223) Type=0.0 Nibs=1
                              + 0D65D SM%MTH(00228) Type=0.0 Nibs=1
sIRAM  =  00002 TI%EQU        -
sIX    =  00007 JT%MTH        -
sInit  =  00003 MB%IMG        -
sKEYS  =  00005 TI%EQU        -
sMAINc =  00005 TI%EQU        -
```

```
sMULT  =  00008 MB%IMG        -
sNEGRD =  0000B JT%MTH        -
sNoChn =  00002 TI%EQU        -
sONERR =  00004 TI%EQU        -
sONTMR =  00006 TI%EQU        -
sPCRD  =  00008 TI%EQU        -
sPRGCF =  0000B FH%TFM        - 16B58 FH%TFM(000AA) Type=0.0 Nibs=1
sRAD   =  00009 SM%MTH        - 0D6DE SM%MTH(002A9) Type=0.0 Nibs=1
                              + 0D7A4 SM%MTH(0036F) Type=0.0 Nibs=1
                              + 0D802 SM%MTH(003CD) Type=0.0 Nibs=1
                              + 0D82C SM%MTH(003F7) Type=0.0 Nibs=1
                              + 0D891 SM%MTH(0045C) Type=0.0 Nibs=1
                              + 0D8C1 SM%MTH(0048C) Type=0.0 Nibs=1
                              + 0DB01 SM%MTH(006CC) Type=0.0 Nibs=1
                              + 0DC0B SM%MTH(007D6) Type=0.0 Nibs=1
                              + 0DC1A SM%MTH(007E5) Type=0.0 Nibs=1
                              + 0DDDD SM%MTH(009A8) Type=0.0 Nibs=1
sRDX   =  0000B MB%IMG        -
sREADI =  00004 TI%EQU        -
sRENAM =  00006 TI%EQU        -
sRENUM =  00008 TI%EQU        -
sRESTR =  0000A TI%EQU        -
sRETRN =  00000 TI%EQU        -
sRFILE =  00008 TI%EQU        -
sRUNBn =  00004 TI%EQU        -
sRUNDC =  00007 TI%EQU        -
sSIGN  =  00009 MB%IMG        -
sSST   =  00002 TI%EQU        -
sSSTdc =  00001 TI%EQU        -
sSTAT  =  00006 TI%EQU        - 0DE3F PM%STA(00010) Type=0.0 Nibs=1
                              + 0DE78 PM%STA(00049) Type=0.0 Nibs=1
                              + 0DEFF PM%STA(000D0) Type=0.0 Nibs=1
                              + 0E023 PM%STA(001F4) Type=0.0 Nibs=1
                              + 0E189 PM%STA(0035A) Type=0.0 Nibs=1
                              + 0E197 PM%STA(00368) Type=0.0 Nibs=1
                              + 0E2CA PM%STA(0049B) Type=0.0 Nibs=1
                              + 0E2D3 PM%STA(004A4) Type=0.0 Nibs=1
                              + 0E2E1 PM%STA(004B2) Type=0.0 Nibs=1
                              + 13483 PM%FLG(0003A) Type=0.0 Nibs=1
                              + 134B1 PM%FLG(00068) Type=0.0 Nibs=1
                              + 1355E PM%FLG(00115) Type=0.0 Nibs=1
                              + 1356A PM%FLG(00121) Type=0.0 Nibs=1
                              + 13620 PM%FLG(001D7) Type=0.0 Nibs=1
                              + 13628 PM%FLG(001DF) Type=0.0 Nibs=1
                              + 136B9 PM%FLG(00270) Type=0.0 Nibs=1
                              + 13758 PM%FLG(0030F) Type=0.0 Nibs=1
sSTOP  =  00005 MB%IMG        -
sSpecl =  00006 MB%IMG        -
sUNDEF =  00001 TI%EQU        - 011D3 TI%UTL(000A2) Type=0.0 Nibs=1
                              + 01239 TI%UTL(00108) Type=0.0 Nibs=1
                              + 16BBB FH%TFM(0010D) Type=0.0 Nibs=1
sXCPT  =  00004 JT%MTH        -
sXQT   =  00000 MB%IMG        -
sXWORD =  00009 TI%EQU        -
savel# =  07390 JP%SYS        - 04F6B SG%LDC(0000D) Type=1.1 Nibs=4 Dist=02425
savel0 =  09702 TI%ERD        - 069FE SG%SYS(0065E) Type=1.1 Nibs=4 Dist=02D04
savlvl =  1CF31 MN%CD:        - 150E3 MN%ED:(00463) Type=1.1 Nibs=4 Dist=07E4E
scrllr =  00171 SB%DVR        -
sflag? =  00194 SB%DVR        - 016CA SB%CMD(000A3) Type=1.1 Nibs=4 Dist=01536
                              + 07B7C JP%SYS(00C4D) Type=1.0 Nibs=4 Dist=079E8
```

```
sflagc =  004CC SB%DVR       - 010AC SB%KEY(003B5) Type=1.1 Nibs=4 Dist=00BE0
                             + 02BFE JP%PR1(0065E) Type=1.1 Nibs=4 Dist=02732
                             + 076E4 JP%SYS(007B5) Type=1.0 Nibs=4 Dist=07218
sflags =  004D3 SB%DVR       - 02613 JP%PR1(00073) Type=1.0 Nibs=4 Dist=02140
                             + 0735C JP%SYS(0042D) Type=1.0 Nibs=4 Dist=06E89
snapr* =  12459 JP%POL       - 0CCE4 JT%MTH(009BD) Type=1.1 Nibs=4 Dist=05775
stkchr =  0A9B4 SB%FCN       - 09367 SG%EXC(00D1F) Type=1.1 Nibs=4 Dist=0164D
                             + 0937D SG%EXC(00D35) Type=1.1 Nibs=4 Dist=01637
stuff  =  16434 AB%BLD       - 11882 SC%FIL(0082F) Type=1.1 Nibs=4 Dist=04BB2

t!     =  000FC SB%TAB       - 02E21 JP%PR2(001FB) Type=0.0 Nibs=2
                             + 052D7 SG%LDC(00379) Type=0.0 Nibs=2
                             + 05418 SG%LDC(004BA) Type=0.0 Nibs=2
                             + 091EE SG%EXC(00BA6) Type=0.0 Nibs=2
                             + 177DC FH%TFM(00D2E) Type=0.0 Nibs=6 Offset=2113280
                             + 19637 SC%SUB(008C5) Type=0.0 Nibs=2
t%     =  00085 SB%TAB       - 1A8A9 SC%REN(0032D) Type=0.0 Nibs=2
                             + 1FE80 AB%LXT(0000C) Type=0.0 Nibs=2
t&     =  00089 SB%TAB       - 15C0D AB%CLC(006EE) Type=0.0 Nibs=2
                             + 1FE83 AB%LXT(0000F) Type=0.0 Nibs=2
t*     =  00083 SB%TAB       - 03AF7 JP%PR3(00721) Type=0.0 Nibs=2
                             + 054D6 SG%LDC(00578) Type=0.0 Nibs=2
                             + 11F41 SC%FIL(00EEE) Type=0.0 Nibs=2
                             + 1FE8F AB%LXT(0001B) Type=0.0 Nibs=2
t+     =  00087 SB%TAB       - 159F2 AB%CLC(004D3) Type=0.0 Nibs=2
                             + 15C3C AB%CLC(0071D) Type=0.0 Nibs=1
                             + 16468 AB%BLD(0013C) Type=0.0 Nibs=1
                             + 1FE92 AB%LXT(0001E) Type=0.0 Nibs=2
t-     =  00082 SB%TAB       - 061EA SB%EXD(008C8) Type=0.0 Nibs=1
                             + 15C30 AB%CLC(00711) Type=0.0 Nibs=2
                             + 16470 AB%BLD(00144) Type=0.0 Nibs=1
                             + 16798 AB%ED:(00126) Type=0.0 Nibs=2
                             + 1FE98 AB%LXT(00024) Type=0.0 Nibs=2
t/     =  00084 SB%TAB       - 1FE9E AB%LXT(0002A) Type=0.0 Nibs=2
t@     =  000F4 SB%TAB       - 02805 JP%PR1(00265) Type=0.0 Nibs=2
                             + 02832 JP%PR1(00292) Type=0.0 Nibs=2
                             + 05410 SG%LDC(004B2) Type=0.0 Nibs=1
                             + 1A8B9 SC%REN(0033D) Type=0.0 Nibs=2
tABS   =  000A2 SB%TAB       -
tACOS  =  0009A SB%TAB       - 1FDE2 SB%KCM(000BE) Type=0.0 Nibs=2
tADD   =  000D5 SB%TAB       - 1FDA4 SB%KCM(00080) Type=0.0 Nibs=2
tADIG0 =  00060 SB%TAB       -
tADIG1 =  00061 SB%TAB       -
tADIG2 =  00062 SB%TAB       -
tADIG3 =  00063 SB%TAB       -
tADIG4 =  00064 SB%TAB       -
tADIG5 =  00065 SB%TAB       -
tADIG6 =  00066 SB%TAB       -
tADIG7 =  00067 SB%TAB       -
tADIG8 =  00068 SB%TAB       -
tADIG9 =  00069 SB%TAB       - 1AA3E SC%REN(004C2) Type=0.0 Nibs=2 Offset=      1
tALL   =  000F8 SB%TAB       - 0331C JP%PR2(006F6) Type=0.0 Nibs=2
                             + 03A32 JP%PR3(0065C) Type=0.0 Nibs=2
                             + 03AA0 JP%PR3(006CA) Type=0.0 Nibs=2
                             + 03ACC JP%PR3(006F6) Type=0.0 Nibs=2
                             + 03C7F JP%PR3(008A9) Type=0.0 Nibs=2
                             + 03EC0 JP%PR3(00AEA) Type=0.0 Nibs=2
                             + 054B1 SG%LDC(00553) Type=0.0 Nibs=2
                             + 05747 SG%LDC(007E9) Type=0.0 Nibs=2
                             + 06458 SG%SYS(000B8) Type=0.0 Nibs=2
```

```
                                + 07644 JP%SYS(00715) Type=0.0 Nibs=2
                                + 0A075 SG%FXQ(004FF) Type=0.0 Nibs=2
                                + 0B239 AB%REG(00641) Type=0.0 Nibs=2
                                + 134DF PM%FLG(00096) Type=0.0 Nibs=2
tAND    =  0008B SB%TAB         -
tANGLE  =  601B3 TI%EQU         - 02D57 JP%PR2(00131) Type=0.0 Nibs=6
                                + 137D0 PM%FLG(00387) Type=0.0 Nibs=6
tARRAY  =  0007D SB%TAB         - 04BE4 AB%LEX(002FE) Type=0.0 Nibs=2
                                + 05387 SG%LDC(00429) Type=0.0 Nibs=2
                                + 0ADBB AB%REG(001C3) Type=0.0 Nibs=2
                                + 164EE AB%BLD(001C2) Type=0.0 Nibs=2
                                + 1AA6C SC%REN(004F0) Type=0.0 Nibs=2
tASIN   =  00099 SB%TAB         - 1FDE0 SB%KCM(000BC) Type=0.0 Nibs=2
tATAN   =  0009B SB%TAB         - 1FDE4 SB%KCM(000C0) Type=0.0 Nibs=2
tAUTO   =  000EE SB%TAB         - 1FDDA SB%KCM(000B6) Type=0.0 Nibs=2
tBASE   =  000E9 SB%TAB         - 02D52 JP%PR2(0012C) Type=0.0 Nibs=2
                                + 05587 SG%LDC(00629) Type=0.0 Nibs=2
                                + 137C7 PM%FLG(0037E) Type=0.0 Nibs=2
tBEEP   =  000E8 SB%TAB         - 1FDC0 SB%KCM(0009C) Type=0.0 Nibs=2
tBIG    =  00010 SB%TAB         - 04E32 AB%LEX(0054C) Type=0.0 Nibs=2
                                + 161C0 AB%CLC(00CA1) Type=0.0 Nibs=2
                                + 1629C AB%CLC(00D7D) Type=0.0 Nibs=2
tCALL   =  000F9 SB%TAB         - 1FDB0 SB%KCM(0008C) Type=0.0 Nibs=2
tCARD   =  000D0 SB%TAB         - 03AA9 JP%PR3(006D3) Type=0.0 Nibs=2
                                + 03B3D JP%PR3(00767) Type=0.0 Nibs=2
                                + 03D43 JP%PR3(0096D) Type=0.0 Nibs=2
                                + 03ECA JP%PR3(00AF4) Type=0.0 Nibs=2
                                + 0576A SG%LDC(0080C) Type=0.0 Nibs=2
                                + 0640D SG%SYS(0006D) Type=0.0 Nibs=2
                                + 08212 JP%EXC(00436) Type=0.0 Nibs=2
                                + 09D75 SG%FXQ(001FF) Type=0.0 Nibs=2
tCAT    =  000EC SB%TAB         - 1FDCE SB%KCM(000AA) Type=0.0 Nibs=2
tCEIL   =  00072 SB%TAB         -
tCFLAG  =  000FA SB%TAB         -
tCHR$   =  000A4 SB%TAB         -
tCLOCK  =  501EF TI%EQU         - 03C6D JP%PR3(00897) Type=0.0 Nibs=6
                                + 05913 SG%LDC(009B5) Type=0.0 Nibs=6
tCMPLX  =  0007A SB%TAB         - 161A5 AB%CLC(00C86) Type=0.0 Nibs=2
tCOLON  =  000E2 SB%TAB         - 0577E SG%LDC(00820) Type=0.0 Nibs=2
                                + 09D84 SG%FXQ(0020E) Type=0.0 Nibs=2
                                + 1A891 SC%REN(00315) Type=0.0 Nibs=2
tCOMMA  =  000F1 SB%TAB         - 03159 JP%PR2(00533) Type=0.0 Nibs=2
                                + 03219 JP%PR2(005F3) Type=0.0 Nibs=2
                                + 03430 JP%PR3(0005A) Type=0.0 Nibs=2
                                + 035B9 JP%PR3(001E3) Type=0.0 Nibs=2
                                + 036D5 JP%PR3(002FF) Type=0.0 Nibs=2
                                + 03BE8 JP%PR3(00812) Type=0.0 Nibs=2
                                + 06A93 SG%SYS(006F3) Type=0.0 Nibs=2
                                + 06BA2 SG%SYS(00802) Type=0.0 Nibs=2
                                + 07136 JP%SYS(00207) Type=0.0 Nibs=2
                                + 08AD6 SG%EXC(0048E) Type=0.0 Nibs=2
                                + 0AEBD AB%REG(002C5) Type=0.0 Nibs=2
                                + 111DB SC%FIL(00188) Type=0.0 Nibs=2
                                + 13D72 SC%DAT(004BD) Type=0.0 Nibs=2
                                + 15C5D AB%CLC(0073E) Type=0.0 Nibs=2
                                + 17CA7 SB%RD:(000E0) Type=0.0 Nibs=2
                                + 17DAB SB%RD:(001E4) Type=0.0 Nibs=2
                                + 17F67 SB%IO:(001AB) Type=0.0 Nibs=2
                                + 18819 SB%IO:(00A5D) Type=0.0 Nibs=2
                                + 188B5 SB%IO:(00AF9) Type=0.0 Nibs=2
```

```
                                + 188DB SB%IO:(00B1F) Type=0.0 Nibs=2
                                + 19E0D SC%SUB(0109B) Type=0.0 Nibs=2
                                + 1A71A SC%REN(0019E) Type=0.0 Nibs=2
                                + 1A8F6 SC%REN(0037A) Type=0.0 Nibs=2
                                + 1FE95 AB%LXT(00021) Type=0.0 Nibs=2
tCOPY   =  000B5 SB%TAB         - 1FDDC SB%KCM(000B8) Type=0.0 Nibs=2
tCOS    =  00097 SB%TAB         - 1FDC6 SB%KCM(000A2) Type=0.0 Nibs=2
tCREF   =  000E0 SB%TAB         - 03973 JP%PR3(0059D) Type=0.0 Nibs=2
tCVAL   =  000E1 SB%TAB         - 03921 JP%PR3(0054B) Type=0.0 Nibs=2
                                + 05CCE SB%EXD(003AC) Type=0.0 Nibs=2
tDATA   =  000C6 SB%TAB         - 17C5A SB%RD:(00093) Type=0.0 Nibs=2
tDATE   =  00077 SB%TAB         -
tDATE$  =  00078 SB%TAB         -
tDEF    =  000B9 SB%TAB         - 03A28 JP%PR3(00652) Type=0.0 Nibs=2
                                + 07C91 JP%SYS(00D62) Type=0.0 Nibs=2
                                + 0FB16 SC%TRC(000CF) Type=0.0 Nibs=2
                                + 19929 SC%SUB(00BB7) Type=0.0 Nibs=2
                                + 1A0E5 SC%SUB(01373) Type=0.0 Nibs=2
                                + 1FDA0 SB%KCM(0007C) Type=0.0 Nibs=2
tDEG    =  0006F SB%TAB         -
tDEGRE  =  000D3 SB%TAB         - 02D72 JP%PR2(0014C) Type=0.0 Nibs=2
                                + 13832 PM%FLG(003E9) Type=0.0 Nibs=2
tDELAY  =  000D6 SB%TAB         -
tDELET  =  000B7 SB%TAB         - 1FDD8 SB%KCM(000B4) Type=0.0 Nibs=2
tDIM    =  000CC SB%TAB         - 1FDBE SB%KCM(0009A) Type=0.0 Nibs=2
tDISP   =  000C5 SB%TAB         - 02968 JP%PR1(003C8) Type=0.0 Nibs=2
                                + 02F42 JP%PR2(0031C) Type=0.0 Nibs=2
                                + 1A7B1 SC%REN(00235) Type=0.0 Nibs=2
                                + 1FDBC SB%KCM(00098) Type=0.0 Nibs=2
tDIV    =  00086 SB%TAB         - 061F5 SB%EXD(008D3) Type=0.0 Nibs=1
                                + 1FF25 AB%LXT(000B1) Type=0.0 Nibs=2
tDMYAR  =  0007E SB%TAB         - 1AA71 SC%REN(004F5) Type=0.0 Nibs=2
                                + 1DBDC TI%FX4(00032) Type=0.0 Nibs=2
tDSTRY  =  000BE SB%TAB         -
tDVZ    =  000B1 SB%TAB         -
tEDIT   =  000B8 SB%TAB         - 1FDCC SB%KCM(000A8) Type=0.0 Nibs=2
tELSE   =  000F5 SB%TAB         - 027EE JP%PR1(0024E) Type=0.0 Nibs=2
                                + 02811 JP%PR1(00271) Type=0.0 Nibs=2
                                + 02A82 JP%PR1(004E2) Type=0.0 Nibs=2
                                + 05318 SG%LDC(003BA) Type=0.0 Nibs=2
                                + 075F0 JP%SYS(006C1) Type=0.0 Nibs=2
                                + 091BA SG%EXC(00B72) Type=0.0 Nibs=2
                                + 1A7BB SC%REN(0023F) Type=0.0 Nibs=2
                                + 1FD98 SB%KCM(00074) Type=0.0 Nibs=2
tEND    =  000DA SB%TAB         -
tENDDF  =  000BA SB%TAB         - 07C96 JP%SYS(00D67) Type=0.0 Nibs=2
                                + 1991A SC%SUB(00BA8) Type=0.0 Nibs=2
tENDSB  =  000C2 SB%TAB         - 07CA0 JP%SYS(00D71) Type=0.0 Nibs=2
tENTER  =  4FFEF TI%EQU         -
tEOL    =  000F0 SB%TAB         - 02670 JP%PR1(000D0) Type=0.0 Nibs=2
                                + 02750 JP%PR1(001B0) Type=0.0 Nibs=2
                                + 02849 JP%PR1(002A9) Type=0.0 Nibs=2
                                + 02A8B JP%PR1(004EB) Type=0.0 Nibs=2
                                + 03834 JP%PR3(0045E) Type=0.0 Nibs=2
                                + 04963 AB%LEX(0007D) Type=0.0 Nibs=2
                                + 04FC9 SG%LDC(0006B) Type=0.0 Nibs=2
                                + 05407 SG%LDC(004A9) Type=0.0 Nibs=2
                                + 06E70 SG%SYS(00AD0) Type=0.0 Nibs=2
                                + 0A59D SG%FXQ(00A27) Type=0.0 Nibs=2
                                + 111D2 SC%FIL(0017F) Type=0.0 Nibs=2
```

```
                                        + 11816 SC%FIL(007C3) Type=0.0 Nibs=2
                                        + 159C8 AB%CLC(004A9) Type=0.0 Nibs=2
                                        + 15C1E AB%CLC(006FF) Type=0.0 Nibs=2
                                        + 16086 AB%CLC(00B67) Type=0.0 Nibs=2
                                        + 162CE AB%CLC(00DAF) Type=0.0 Nibs=2
                                        + 164A6 AB%BLD(0017A) Type=0.0 Nibs=2
                                        + 1741F FH%TFM(00971) Type=0.0 Nibs=2
                                        + 17478 FH%TFM(009CA) Type=0.0 Nibs=2
                                        + 177EB FH%TFM(00D3D) Type=0.0 Nibs=2
                                        + 17CB0 SB%RD:(000E9) Type=0.0 Nibs=1
                                        + 18822 SB%IO:(00A66) Type=0.0 Nibs=1
                                        + 188E4 SB%IO:(00B28) Type=0.0 Nibs=1
                                        + 18DB6 SC%SUB(00044) Type=0.0 Nibs=2
                                        + 1ADCD SB%VAL(0005F) Type=0.0 Nibs=2
                                        + 1ADE0 SB%VAL(00072) Type=0.0 Nibs=2
                                        + 1B491 MB%IMG(0004B) Type=0.0 Nibs=1
                                        + 1C10A MB%USG(00666) Type=0.0 Nibs=2
tEPS    =  00071 SB%TAB         -
tERRL   =  00075 SB%TAB         -
tERRN   =  00076 SB%TAB         -
tERROR  =  000E3 SB%TAB         - 02B2C JP%PR1(0058C) Type=0.0 Nibs=2
                                        + 03369 JP%PR2(00743) Type=0.0 Nibs=2
                                        + 05552 SG%LDC(005F4) Type=0.0 Nibs=2
                                        + 07E39 JP%EXC(0005D) Type=0.0 Nibs=2
                                        + 07F98 JP%EXC(001BC) Type=0.0 Nibs=2
                                        + 1A7CF SC%REN(00253) Type=0.0 Nibs=2
tEXOR   =  0008C SB%TAB         -
tEXP    =  00094 SB%TAB         - 1FDCA SB%KCM(000A6) Type=0.0 Nibs=2
tEXTIF  =  000F4 SB%TAB         - 034EC JP%PR3(00116) Type=0.0 Nibs=2
                                        + 091E9 SG%EXC(00BA1) Type=0.0 Nibs=2
                                        + 1A833 SC%REN(002B7) Type=0.0 Nibs=2
tEXTND  =  601EF TI%EQU         - 03CB2 JP%PR3(008DC) Type=0.0 Nibs=6
tFACT   =  000A8 SB%TAB         - 1FDC2 SB%KCM(0009E) Type=0.0 Nibs=2
tFETCH  =  000C8 SB%TAB         - 1FDD4 SB%KCM(000B0) Type=0.0 Nibs=2
tFFN    =  000B4 SB%TAB         -
tFLOW   =  901EF TI%EQU         - 03347 JP%PR2(00721) Type=0.0 Nibs=6
                                        + 0FA6A SC%TRC(00023) Type=0.0 Nibs=6
tFLT1   =  0001D SB%TAB         -
tFLT10  =  00014 SB%TAB         -
tFLT11  =  00013 SB%TAB         -
tFLT12  =  00012 SB%TAB         -
tFLT2   =  0001C SB%TAB         -
tFLT3   =  0001B SB%TAB         -
tFLT4   =  0001A SB%TAB         -
tFLT5   =  00019 SB%TAB         -
tFLT6   =  00018 SB%TAB         -
tFLT7   =  00017 SB%TAB         -
tFLT8   =  00016 SB%TAB         -
tFLT9   =  00015 SB%TAB         -
tFN     =  0007C SB%TAB         - 02900 JP%PR1(00360) Type=0.0 Nibs=2
                                        + 02AE3 JP%PR1(00543) Type=0.0 Nibs=2
                                        + 030A0 JP%PR2(0047A) Type=0.0 Nibs=2
                                        + 034BF JP%PR3(000E9) Type=0.0 Nibs=2
                                        + 04B85 AB%LEX(0029F) Type=0.0 Nibs=2
                                        + 051DE SG%LDC(00280) Type=0.0 Nibs=2
                                        + 15AB0 AB%CLC(00591) Type=0.0 Nibs=2
                                        + 164F7 AB%BLD(001CB) Type=0.0 Nibs=1
                                        + 1AA67 SC%REN(004EB) Type=0.0 Nibs=2
tFOR    =  000C3 SB%TAB         - 1FD9A SB%KCM(00076) Type=0.0 Nibs=2
tFP     =  0006B SB%TAB         -
```

```
tGOSUB =  000DC SB%TAB        - 02B7F JP%PR1(005DF) Type=0.0 Nibs=2
                              + 04C79 AB%LEX(00393) Type=0.0 Nibs=2
                              + 1A798 SC%REN(0021C) Type=0.0 Nibs=2
                              + 1FDB2 SB%KCM(0008E) Type=0.0 Nibs=2
tGOTO  =  000DD SB%TAB        - 02B84 JP%PR1(005E4) Type=0.0 Nibs=2
                              + 04C80 AB%LEX(0039A) Type=0.0 Nibs=2
                              + 07EFA JP%EXC(0011E) Type=0.0 Nibs=1
                              + 08027 JP%EXC(0024B) Type=0.0 Nibs=1
                              + 1A79D SC%REN(00221) Type=0.0 Nibs=2
                              + 1FDB6 SB%KCM(00092) Type=0.0 Nibs=2
tIF    =  000DF SB%TAB        - 027A4 JP%PR1(00204) Type=0.0 Nibs=2
                              + 1A7B6 SC%REN(0023A) Type=0.0 Nibs=2
                              + 1FD94 SB%KCM(00070) Type=0.0 Nibs=2
tIMAGE =  000FF SB%TAB        -
tIN    =  000F2 SB%TAB        - 0394F JP%PR3(00579) Type=0.0 Nibs=2
                              + 058D3 SG%LDC(00975) Type=0.0 Nibs=2
                              + 190E3 SC%SUB(00371) Type=0.0 Nibs=2
tINF   =  00070 SB%TAB        - 046E6 SB%EXP(00711) Type=0.0 Nibs=2
tINPUT =  000C9 SB%TAB        - 1FDB8 SB%KCM(00094) Type=0.0 Nibs=2
tINT   =  0009C SB%TAB        -
tINT10 =  00004 SB%TAB        -
tINT11 =  00003 SB%TAB        -
tINT12 =  00002 SB%TAB        -
tINT2  =  0000C SB%TAB        - 0385B JP%PR3(00485) Type=0.0 Nibs=2
tINT3  =  0000B SB%TAB        -
tINT4  =  0000A SB%TAB        -
tINT5  =  00009 SB%TAB        -
tINT6  =  00008 SB%TAB        -
tINT7  =  00007 SB%TAB        -
tINT8  =  00006 SB%TAB        -
tINT9  =  00005 SB%TAB        -
tINTEG =  000CA SB%TAB        -
tINTO  =  E01EF TI%EQU        - 03E45 JP%PR3(00A6F) Type=0.0 Nibs=6
                              + 03E61 JP%PR3(00A8B) Type=0.0 Nibs=6
                              + 03ECF JP%PR3(00AF9) Type=0.0 Nibs=6
                              + 05634 SG%LDC(006D6) Type=0.0 Nibs=6
                              + 16ACA FH%TFM(0001C) Type=0.0 Nibs=6
tINTR  =  015FF TI%EQU        - 1A855 SC%REN(002D9) Type=0.0 Nibs=5
tINX   =  000B2 SB%TAB        -
tIP    =  0006A SB%TAB        -
tIS    =  000E7 SB%TAB        -
tISUB$ =  000A7 SB%TAB        - 1AA76 SC%REN(004FA) Type=0.0 Nibs=2
tIVL   =  000AE SB%TAB        -
tKEY   =  000E5 SB%TAB        - 03097 JP%PR2(00471) Type=0.0 Nibs=2
                              + 03C21 JP%PR3(0084B) Type=0.0 Nibs=2
                              + 0739E JP%SYS(0046F) Type=0.0 Nibs=2
                              + 1FDA2 SB%KCM(0007E) Type=0.0 Nibs=2
tKEY$  =  00073 SB%TAB        -
tKEYS  =  000CF SB%TAB        - 03B63 JP%PR3(0078D) Type=0.0 Nibs=2
                              + 03BA7 JP%PR3(007D1) Type=0.0 Nibs=2
                              + 03EC5 JP%PR3(00AEF) Type=0.0 Nibs=2
                              + 05765 SG%LDC(00807) Type=0.0 Nibs=2
                              + 0A362 SG%FXQ(007EC) Type=0.0 Nibs=2
tKYSck =  0A35D SG%FXQ        - 06468 SG%SYS(000C8) Type=1.1 Nibs=4 Dist=03EF5
                              + 06A9C SG%SYS(006FC) Type=1.1 Nibs=4 Dist=038C1
                              + 08230 JP%EXC(00454) Type=1.1 Nibs=4 Dist=0212D
tLBLRF =  0000E SB%TAB        - 02A31 JP%PR1(00491) Type=0.0 Nibs=2
                              + 052CD SG%LDC(0036F) Type=0.0 Nibs=2
                              + 07334 JP%SYS(00405) Type=0.0 Nibs=2
tLBLST =  000F6 SB%TAB        - 02727 JP%PR1(00187) Type=0.0 Nibs=2
```

```
                                + 04FAA SG%LDC(0004C) Type=0.0 Nibs=2
                                + 0779E JP%SYS(0086F) Type=0.0 Nibs=2
                                + 07832 JP%SYS(00903) Type=0.0 Nibs=2
                                + 07C8C JP%SYS(00D5D) Type=0.0 Nibs=2
                                + 1B4B1 MB%IMG(0006B) Type=0.0 Nibs=2
tLEN    =  000A9 SB%TAB         -
tLET    =  000C0 SB%TAB         -
tLINE#  =  0000F SB%TAB         - 029E8 JP%PR1(00448) Type=0.0 Nibs=2
                                + 034C4 JP%PR3(000EE) Type=0.0 Nibs=2
                                + 03BDF JP%PR3(00809) Type=0.0 Nibs=2
                                + 0510B SG%LDC(001AD) Type=0.0 Nibs=2
                                + 052D2 SG%LDC(00374) Type=0.0 Nibs=2
                                + 07145 JP%SYS(00216) Type=0.0 Nibs=2
                                + 073CD JP%SYS(0049E) Type=0.0 Nibs=2
                                + 07A9E JP%SYS(00B6F) Type=0.0 Nibs=2
                                + 07F1E JP%EXC(00142) Type=0.0 Nibs=2
                                + 1A818 SC%REN(0029C) Type=0.0 Nibs=2
                                + 1A8CF SC%REN(00353) Type=0.0 Nibs=2
                                + 1B44E MB%IMG(00008) Type=0.0 Nibs=2
tLINPT  =  000BF SB%TAB         - 18790 SB%IO:(009D4) Type=0.0 Nibs=2
tLIST   =  000BB SB%TAB         - 1FDD6 SB%KCM(000B2) Type=0.0 Nibs=2
tLITRL  =  000C4 SB%TAB         - 03CEB JP%PR3(00915) Type=0.0 Nibs=2
                                + 05708 SG%LDC(007AA) Type=0.0 Nibs=2
                                + 05760 SG%LDC(00802) Type=0.0 Nibs=2
                                + 07F3D JP%EXC(00161) Type=0.0 Nibs=2
                                + 09D8E SG%FXQ(00218) Type=0.0 Nibs=2
                                + 1A8A0 SC%REN(00324) Type=0.0 Nibs=2
                                + 1A8D8 SC%REN(0035C) Type=0.0 Nibs=2
tLN     =  00091 SB%TAB         -
tLOG    =  00090 SB%TAB         - 1FDE6 SB%KCM(000C2) Type=0.0 Nibs=2
tLOG10  =  00093 SB%TAB         -
tLPRP   =  000AA SB%TAB         - 043A1 SB%EXP(003CC) Type=0.0 Nibs=2
tLR     =  000B6 SB%TAB         - 1FDA6 SB%KCM(00082) Type=0.0 Nibs=2
tMAIN   =  000D2 SB%TAB         - 03D39 JP%PR3(00963) Type=0.0 Nibs=2
                                + 05774 SG%LDC(00816) Type=0.0 Nibs=2
                                + 09D7F SG%FXQ(00209) Type=0.0 Nibs=2
tMATH   =  601EF TI%EQU         - 03C84 JP%PR3(008AE) Type=0.0 Nibs=6
                                + 054BA SG%LDC(0055C) Type=0.0 Nibs=6
                                + 134F8 PM%FLG(000AF) Type=0.0 Nibs=6
tMAX    =  000AD SB%TAB         -
tMAXRL  =  0006C SB%TAB         - 046ED SB%EXP(00718) Type=0.0 Nibs=2
tMEAN   =  0009D SB%TAB         - 1FDAA SB%KCM(00086) Type=0.0 Nibs=2
tMIN    =  000AC SB%TAB         -
tMOD    =  00074 SB%TAB         -
tNAME   =  000BD SB%TAB         - 1FDD0 SB%KCM(000AC) Type=0.0 Nibs=2
tNEAR   =  C01EF TI%EQU         - 02D85 JP%PR2(0015F) Type=0.0 Nibs=6
tNEG    =  D01EF TI%EQU         - 02D97 JP%PR2(00171) Type=0.0 Nibs=6
                                + 13804 PM%FLG(003BB) Type=0.0 Nibs=6
tNEXT   =  000C4 SB%TAB         - 087DE SG%EXC(00196) Type=0.0 Nibs=2
                                + 1FD9E SB%KCM(0007A) Type=0.0 Nibs=2
tNOT    =  00081 SB%TAB         - 061E2 SB%EXD(008C0) Type=0.0 Nibs=1
                                + 15C16 AB%CLC(006F7) Type=0.0 Nibs=1
tNUM    =  000A3 SB%TAB         -
tOFF    =  000E1 SB%TAB         - 02ABB JP%PR1(0051B) Type=0.0 Nibs=2
                                + 03339 JP%PR2(00713) Type=0.0 Nibs=2
                                + 03CAD JP%PR3(008D7) Type=0.0 Nibs=2
                                + 03DA3 JP%PR3(009CD) Type=0.0 Nibs=2
                                + 05855 SG%LDC(008F7) Type=0.0 Nibs=2
                                + 13792 PM%FLG(00349) Type=0.0 Nibs=2
tON     =  000E0 SB%TAB         - 02AB6 JP%PR1(00516) Type=0.0 Nibs=2
```

```
                                + 03CA8 JP%PR3(008D2) Type=0.0 Nibs=2
                                + 03D9E JP%PR3(009C8) Type=0.0 Nibs=2
                                + 05850 SG%LDC(008F2) Type=0.0 Nibs=2
                                + 07E00 JP%EXC(00024) Type=0.0 Nibs=2
                                + 0EA73 MN%BP:(00082) Type=0.0 Nibs=2
                                + 1377F PM%FLG(00336) Type=0.0 Nibs=2
                                + 1A7A7 SC%REN(0022B) Type=0.0 Nibs=2
tOPT'N =  000ED SB%TAB          -
tOR    =  0008D SB%TAB          -
tOVF   =  000AF SB%TAB          -
tPAUSE =  000D7 SB%TAB          -
tPCRD  =  E01EF TI%EQU          - 03D48 JP%PR3(00972) Type=0.0 Nibs=6
                                + 05779 SG%LDC(0081B) Type=0.0 Nibs=2
                                + 0579D SG%LDC(0083F) Type=0.0 Nibs=6
                                + 09CFE SG%FXQ(00188) Type=0.0 Nibs=6
tPI    =  00079 SB%TAB          -
tPORT  =  000D1 SB%TAB          - 033A0 JP%PR2(0077A) Type=0.0 Nibs=2
                                + 03D3E JP%PR3(00968) Type=0.0 Nibs=2
                                + 0576F SG%LDC(00811) Type=0.0 Nibs=2
                                + 09D7A SG%FXQ(00204) Type=0.0 Nibs=2
tPOS   =  201B3 TI%EQU          - 02D8E JP%PR2(00168) Type=0.0 Nibs=6
                                + 137EA PM%FLG(003A1) Type=0.0 Nibs=6
tPREDV =  0009F SB%TAB          - 1FDA8 SB%KCM(00084) Type=0.0 Nibs=2
tPRINT =  000CD SB%TAB          - 1A7AC SC%REN(00230) Type=0.0 Nibs=2
                                + 1FDBA SB%KCM(00096) Type=0.0 Nibs=2
tPRMEN =  000F8 SB%TAB          - 03821 JP%PR3(0044B) Type=0.0 Nibs=2
                                + 03936 JP%PR3(00560) Type=0.0 Nibs=2
                                + 05691 SG%LDC(00733) Type=0.0 Nibs=2
                                + 18E49 SC%SUB(000D7) Type=0.0 Nibs=2
                                + 1922C SC%SUB(004BA) Type=0.0 Nibs=2
tPRMST =  000F3 SB%TAB          - 030E3 JP%PR2(004BD) Type=0.0 Nibs=2
                                + 039BC JP%PR3(005E6) Type=0.0 Nibs=2
                                + 05239 SG%LDC(002DB) Type=0.0 Nibs=2
                                + 05673 SG%LDC(00715) Type=0.0 Nibs=2
                                + 18E3A SC%SUB(000C8) Type=0.0 Nibs=2
                                + 19ACF SC%SUB(00D5D) Type=0.0 Nibs=2
tPURGE =  000EB SB%TAB          - 1FDD2 SB%KCM(000AE) Type=0.0 Nibs=2
tRAD   =  0006E SB%TAB          -
tRDIAN =  000D4 SB%TAB          - 02D77 JP%PR2(00151) Type=0.0 Nibs=2
tREAD  =  000C7 SB%TAB          -
tREAL  =  000BC SB%TAB          -
tRELOP =  0008A SB%TAB          - 04A1B AB%LEX(00135) Type=0.0 Nibs=2
                                + 059CD SB%EXD(000AB) Type=0.0 Nibs=2 Offset=   -112
                                + 05D92 SB%EXD(00470) Type=0.0 Nibs=2
                                + 061FD SB%EXD(008DB) Type=0.0 Nibs=1
                                + 15F35 AB%CLC(00A16) Type=0.0 Nibs=2
                                + 16478 AB%BLD(0014C) Type=0.0 Nibs=1
                                + 1AA47 SC%REN(004CB) Type=0.0 Nibs=2
tREM   =  000E6 SB%TAB          -
tRES   =  0007F SB%TAB          - 15A66 AB%CLC(00547) Type=0.0 Nibs=2
                                + 1FDDE SB%KCM(000BA) Type=0.0 Nibs=2
                                + 1FDFA SB%KCM(000D6) Type=0.0 Nibs=2
tRESTR =  000DE SB%TAB          - 02B89 JP%PR1(005E9) Type=0.0 Nibs=2
                                + 1A7A2 SC%REN(00226) Type=0.0 Nibs=2
tRETRN =  000DB SB%TAB          - 1FDB4 SB%KCM(00090) Type=0.0 Nibs=2
tRFILE =  000DE SB%TAB          - 03DB3 JP%PR3(009DD) Type=0.0 Nibs=2
                                + 0550D SG%LDC(005AF) Type=0.0 Nibs=2
                                + 06F73 JP%SYS(00044) Type=0.0 Nibs=2
tRMD   =  0006D SB%TAB          -
tRND   =  000A0 SB%TAB          -
```

```
tROUND =  C01EF TI%EQU       - 02D60 JP%PR2(0013A) Type=0.0 Nibs=6
tRUN   =  000FE SB%TAB       -
tSDEV  =  0009E SB%TAB       - 1FDAC SB%KCM(00088) Type=0.0 Nibs=2
tSEMIC =  000F2 SB%TAB       - 031CD JP%PR2(005A7) Type=0.0 Nibs=2
                             + 031E2 JP%PR2(005BC) Type=0.0 Nibs=2
                             + 03214 JP%PR2(005EE) Type=0.0 Nibs=2
                             + 0322C JP%PR2(00606) Type=0.0 Nibs=2
                             + 032F4 JP%PR2(006CE) Type=0.0 Nibs=2
                             + 033F2 JP%PR3(0001C) Type=0.0 Nibs=2
                             + 03408 JP%PR3(00032) Type=0.0 Nibs=2
                             + 035BE JP%PR3(001E8) Type=0.0 Nibs=2
                             + 0365B JP%PR3(00285) Type=0.0 Nibs=2
                             + 03775 JP%PR3(0039F) Type=0.0 Nibs=2
                             + 05167 SG%LDC(00209) Type=0.0 Nibs=2
                             + 14660 SC%DAT(00DAB) Type=0.0 Nibs=2
                             + 17F62 SB%IO:(001A6) Type=0.0 Nibs=2
                             + 186C6 SB%IO:(0090A) Type=0.0 Nibs=2
                             + 19A8E SC%SUB(00D1C) Type=0.0 Nibs=2
                             + 1FEC2 AB%LXT(0004E) Type=0.0 Nibs=2
tSFLAG =  000FB SB%TAB       -
tSGN   =  000A1 SB%TAB       -
tSHORT =  000CB SB%TAB       -
tSIN   =  00096 SB%TAB       - 1FDC4 SB%KCM(000A0) Type=0.0 Nibs=2
tSMALL =  00011 SB%TAB       - 04655 SB%EXP(00680) Type=0.0 Nibs=2
                             + 1629A AB%CLC(00D7B) Type=0.0 Nibs=2
tSQR   =  00092 SB%TAB       - 1FDAE SB%KCM(0008A) Type=0.0 Nibs=2
tSTAT  =  000CE SB%TAB       -
tSTEP  =  000F6 SB%TAB       - 0344B JP%PR3(00075) Type=0.0 Nibs=2
                             + 086D2 SG%EXC(0008A) Type=0.0 Nibs=2
tSTOP  =  000D9 SB%TAB       -
tSTR$  =  000A6 SB%TAB       -
tSUB   =  000C1 SB%TAB       - 03A2D JP%PR3(00657) Type=0.0 Nibs=2
                             + 07BEF JP%SYS(00CC0) Type=0.0 Nibs=2
                             + 07C9B JP%SYS(00D6C) Type=0.0 Nibs=2
                             + 1A35B SC%SUB(015E9) Type=0.0 Nibs=2
tSVAR  =  0002D SB%TAB       - 031A6 JP%PR2(00580) Type=0.0 Nibs=2
                             + 0539E SG%LDC(00440) Type=0.0 Nibs=2
                             + 05FC9 SB%EXD(006A7) Type=0.0 Nibs=2
tTAB   =  000F7 SB%TAB       - 035C3 JP%PR3(001ED) Type=0.0 Nibs=2
                             + 05484 SG%LDC(00526) Type=0.0 Nibs=1
                             + 17F6C SB%IO:(001B0) Type=0.0 Nibs=2
tTAN   =  00098 SB%TAB       - 1FDC8 SB%KCM(000A4) Type=0.0 Nibs=2
tTHEN  =  000F4 SB%TAB       - 03483 JP%PR3(000AD) Type=0.0 Nibs=2
                             + 05305 SG%LDC(003A7) Type=0.0 Nibs=2
                             + 1FD96 SB%KCM(00072) Type=0.0 Nibs=2
tTIME  =  0007B SB%TAB       -
tTIME$ =  00095 SB%TAB       -
tTIMER =  000E4 SB%TAB       - 02B31 JP%PR1(00591) Type=0.0 Nibs=2
                             + 0336E JP%PR2(00748) Type=0.0 Nibs=2
tTO    =  000F3 SB%TAB       - 0343A JP%PR3(00064) Type=0.0 Nibs=2
                             + 03AE8 JP%PR3(00712) Type=0.0 Nibs=2
                             + 03B2F JP%PR3(00759) Type=0.0 Nibs=2
                             + 03B51 JP%PR3(0077B) Type=0.0 Nibs=2
                             + 03EBB JP%PR3(00AE5) Type=0.0 Nibs=2
                             + 05728 SG%LDC(007CA) Type=0.0 Nibs=2
                             + 081E7 JP%EXC(0040B) Type=0.0 Nibs=2
                             + 1FD9C SB%KCM(00078) Type=0.0 Nibs=2
tTRACE =  000EA SB%TAB       -
tUNF   =  000B0 SB%TAB       -
tUPRC$ =  000AB SB%TAB       -
```

```
tUSER  =  000E2 SB%TAB        -
tUSING =  000FD SB%TAB        - 0362C JP%PR3(00256) Type=0.0 Nibs=2
                              + 05452 SG%LDC(004F4) Type=0.0 Nibs=2
                              + 17F40 SB%IO:(00184) Type=0.0 Nibs=2
                              + 1A806 SC%REN(0028A) Type=0.0 Nibs=2
tVAL   =  000A5 SB%TAB        -
tVARS  =  B01EF TI%EQU        - 0333E JP%PR2(00718) Type=0.0 Nibs=6
                              + 0FA56 SC%TRC(0000F) Type=0.0 Nibs=6
tWAIT  =  000D8 SB%TAB        -
tXFN   =  000B3 SB%TAB        - 02C84 JP%PR2(0005E) Type=0.0 Nibs=2
                              + 02CCF JP%PR2(000A9) Type=0.0 Nibs=2
                              + 04BA8 AB%LEX(002C2) Type=0.0 Nibs=2
                              + 05084 SG%LDC(00126) Type=0.0 Nibs=2
                              + 15F23 AB%CLC(00A04) Type=0.0 Nibs=2
                              + 1AA62 SC%REN(004E6) Type=0.0 Nibs=2
tXWORD =  000EF SB%TAB        - 027AD JP%PR1(0020D) Type=0.0 Nibs=2
                              + 02C4C JP%PR2(00026) Type=0.0 Nibs=2
                              + 02CC6 JP%PR2(000A0) Type=0.0 Nibs=2
                              + 04B24 AB%LEX(0023E) Type=0.0 Nibs=2
                              + 0507B SG%LDC(0011D) Type=0.0 Nibs=2
                              + 09D89 SG%FXQ(00213) Type=0.0 Nibs=2
                              + 15F2C AB%CLC(00A0D) Type=0.0 Nibs=2
                              + 1A7C0 SC%REN(00244) Type=0.0 Nibs=2
tZ     =  0005A SB%TAB        - 1AA29 SC%REN(004AD) Type=0.0 Nibs=2
tZERO  =  C01EF TI%EQU        - 02DA0 JP%PR2(0017A) Type=0.0 Nibs=6
                              + 137F7 PM%FLG(003AE) Type=0.0 Nibs=6
t^     =  00080 SB%TAB        - 15A18 AB%CLC(004F9) Type=0.0 Nibs=2
                              + 1FF2B AB%LXT(000B7) Type=0.0 Nibs=2

uAD12  =  0C922 JT%MTH        - 0889C SG%EXC(00254) Type=1.1 Nibs=4 Dist=04086
                              + 0B6C5 AB%FCN(0008C) Type=1.1 Nibs=4 Dist=0125D
uALit  =  000F7 MB%IMG        -
uCPLXC =  000EE MB%IMG        -
uDELIM =  000F4 MB%IMG        -
uHKB^  =  000F6 MB%IMG        -
uIMXCH =  000D4 MB%IMG        -
uIMbck =  000DC MB%IMG        -
uIMend =  000F0 MB%IMG        -
uIMsta =  000DE MB%IMG        -
uJMPdl =  000DB MB%IMG        -
uJMPst =  000DA MB%IMG        -
uJMP{} =  000D9 MB%IMG        -
uLOOPB =  000D2 MB%IMG        -
uLOOPP =  000EF MB%IMG        -
uLOOPS =  000D3 MB%IMG        -
uMODES =  0BDB1 AB%FCN        - 0C924 JT%MTH(005FD) Type=1.1 Nibs=4 Dist=00B73
                              + 0C9D1 JT%MTH(006AA) Type=1.1 Nibs=4 Dist=00C20
                              + 0E224 PM%STA(003F5) Type=1.0 Nibs=4 Dist=02473
uMULT  =  000D1 MB%IMG        -
uNUMEn =  000FC MB%IMG        -
uNUMEs =  000FD MB%IMG        -
uNUMFn =  000FA MB%IMG        -
uNUMFs =  000FB MB%IMG        -
uNUMNn =  000F8 MB%IMG        -
uNUMNs =  000F9 MB%IMG        -
uOPNM- =  000DF MB%IMG        -
uOPNNM =  000D8 MB%IMG        -
uOPNWM =  000E0 MB%IMG        -
uRES12 =  0C994 JT%MTH        - 0BC80 AB%FCN(00647) Type=1.0 Nibs=4 Dist=00D14
                              + 0E1CC PM%STA(0039D) Type=1.0 Nibs=4 Dist=01838
```

```
uRESD1 =  0E1EE PM%STA        -
uRESNX =  0C9BD JT%MTH        - 0F97E AB%ASN(003A4) Type=1.1 Nibs=4 Dist=02FC1
uRESTP =  000F1 MB%IMG        -
uRESXT =  0C9C1 JT%MTH        - 0D470 SM%MTH(0003B) Type=1.1 Nibs=4 Dist=00AAF
                              + 0F9E3 AB%ASN(00409) Type=1.1 Nibs=4 Dist=03022
uRND12 =  0C9CD JT%MTH        -
uRND>P =  0C9CF JT%MTH        -
uSTRPT =  000D0 MB%IMG        -
uTEST  =  0D435 SM%MTH        - 0B7F4 AB%FCN(001BB) Type=1.1 Nibs=4 Dist=01C41
ufCHN# =  00067 TI%EQU        -
ufCNTA =  00010 TI%EQU        -
ufFSTK =  00053 TI%EQU        -
ufGSTK =  00058 TI%EQU        -
ufLENG =  0006A TI%EQU        -
ufMSTK =  0004E TI%EQU        -
ufPC   =  0000B TI%EQU        -
ufPMCN =  0005D TI%EQU        -
ufPMTR =  0005F TI%EQU        -
ufRSTK =  0001A TI%EQU        -
ufRTNA =  00006 TI%EQU        -
ufRTNT =  00069 TI%EQU        -
ufSR-0 =  0002E TI%EQU        -
ufSR-1 =  0003E TI%EQU        -
ufSTM0 =  00015 TI%EQU        -
ufSTM1 =  00029 TI%EQU        -
ufSTSV =  00064 TI%EQU        -

viewd1 =  00178 SB%DVR        -

waitky =  1517D MN%ED:        - 15959 AB%CLC(0043A) Type=1.1 Nibs=3 Dist=007DC
warm1x =  0018A SB%DVR        -

xANGLE =  00006 JP%TAB        -
xCLOCK =  00015 JP%TAB        -
xEXTND =  00026 JP%TAB        -
xFLOW  =  00029 JP%TAB        -
xINTO  =  0002E JP%TAB        -
xMATH  =  00036 JP%TAB        -
xNEAR  =  0003C JP%TAB        -
xNEG   =  0003D JP%TAB        -
xPCRD  =  0003E JP%TAB        -
xPOS   =  00042 JP%TAB        -
xROUND =  0004C JP%TAB        -
xVARS  =  0005B JP%TAB        -
xZERO  =  0001C JP%TAB        -
xrm01s =  1DD04 JP%TAB        - 10E44 MN%CNF(00CB2) Type=0.0 Nibs=5
xrom01 =  1DD6C JP%TAB        -
```

Saturn Hex Code Listing

```
00000 - 2034EE10 0060F481 2C1361B0 04F21547 19011507 1902AF41 507AF207 80F62545
00040 - 0B0627A0 E04B0616 F15C7838 E1828838 207D9685 10B15C04 60682120 1B074F21
00080 - 5E090AB0 D215C05C 4198315A 333F8612 39126069 5F163146 8AA60060 31926AF2
000C0 - 15CD1A20 1E14E0B8 400B14C7 E501B034 F215E706 2790A400 525A0E80 F6190215

00100 - 67AF5190 11527190 01567134 12C0F8DC B9108D89 8108DC98 108D99C4 18D7FC00
00140 - 8DE3C108 D344208D ADA108DD 4D008D15 7108DEE0 108DAB25 18DE2120 8D74151D
00180 - 98058CF4 578CA1F0 6C358DC4 63180AAF 2203081B FF1E215C 01868288 38813068
001C0 - F2935130 18018075 4E7F7F8F F0201343 93900631 91260F1F 834F233F 86115D38

00200 - 28AF01DC 3D222308 8FFA0B17 C2F04744 41A649F3 3400115C 31967154 08508F21
00240 - 2018F121 3178048F CCDE08F9 84011D49 143131E2 8FFA0B18 E237284D 84E1FD87
00280 - F2AF2A7E 15DD17D1 5DD1EF49 F36064D0 A014D178 15D81FF8 6008E559 185B8F33
002C0 - 5A020323 08D1E5E5 8FD7911D 214D1AEF 6F3E3353 80333005 99915CE7 721FF207

00300 - CA35606F 128EE191 4037B01A F8EEA824 818C5A66 D0A0B1C3 E3B1E3FF 8E0F8159
00340 - 08DEE851 8ED83731 5D7D7131 6E763E4F 175718EB FE190AA0 7DFD7261 8E878178
00380 - 5184E8FB F1908E6B 01316E72 318F7E09 0322088F 14A11315 D7E11797 D5E6D230
003C0 - DEA101CE 96241325 088EFF14 56079E03 15D75E07 5218F7D1 511198EB 202EA05E

00400 - 0EA01B04 B0D70080 380680DC 08A0A010 808CCF17 315D7C90 74901371 08113D8E
00440 - 5322088F D7911503 11013210 011BCE8F 771B131D 014D1101 31175137 8CCA1222
00480 - 8DF44908 CF5218C0 8218C192 18C1A218 E691168D E8C57128 C69A68C9 8A68CB6D
004C0 - 68DE8851 311D8D10 6318DAF5 318EDC17 4008E49F 18E4A618 75908FA3 A818FA8C

00500 - 418EF942 740060FD 1F244F2D 01532159 0A2C0104 2031FC71 AF734032 9088FAB8
00540 - 11590137 6F2F3280 88FAB811 4CE61ADB 1E3D0A0B 1C3FF042 031FC735 F208F231
00580 - 211FF550 08E11618 65501778 E1761718 ECF31DC7 82F31EC7 91F795F1 C01491EF
005C0 - F3E15901 E201E159 01DFF159 08EA9C58 078FF020 12032908 8F14A118 FD75217E

00600 - 0F53386D 41843099 0AA031DC 70BE720E EF2031DC 786B5907 DDE4928E 14E12031
00640 - 4C768E78 DDDF8F89 C815D08D 92FE06F3 F602F1B0 53E23F80 01400220 04100815
00680 - 4719EF31 9114C03B 1C3D456D 6F627970 2C4F6374 7D0A0FF3 1EC7DDA4 0031DC65
006C0 - DA713678 405008EE AB5D2803 8AE00722 08410B15 C0807838 6068AA79 008715F0

00700 - B021B3E6 F215E00B 011B344F 21522630 E8F71FE0 8E56411B DF3E28F6 8FE05808
00740 - E99318FD 75218450 98F94A21 87CD080E 834B0824 76BC9F8C 01410680 3F2C6C6C
00780 - E070124F 7F7F7F7F 780C1E3F 7C100000 00000007 05070005 49211925 480C1A28
007C0 - 08083444 48344CFA 0A4A443F 71010103 003C3E7C 303F151A 08586834 4C44340F

00800 - 08000878 217A0408 007CF040 4C304020 78A02F38 040C7402 08055F75 580E3949
00840 - 494E3C52 62026C50 3A4D4940 30083454 40040C74 0C740C73 12131C70 25545558
00880 - 7C334243 4C383544 45483C31 40414C3C 31404D30 4F151170 50514365 59436414
008C0 - 3C161418 4E794142 055A255A 25500000 00000000 0F500000 07000700 041F741F

00900 - 74142A2F 7A221323 18046266 39465020 50000700 00000C12 21400001 422C1008
00940 - 0A2C1A28 08080E38 080000B0 70000808 08080000 00606000 00201804 020E3159
00980 - 454E3002 4F704002 61594946 42294949 46381412 1F701725 4545493C 3A494940
009C0 - 31017905 03063949 49463609 49492E10 06363000 0006B670 00080412 21400414

00A00 - 14141411 42241800 02010159 060E314D 594E4E79 09090E7F 79494946 3E314141
00A40 - 422F7141 422C1F79 4949414F 79090901 0E314141 527F7808 080F7001 4F714000
00A80 - 3040404F 3F780412 214F7040 40404F72 0C020F7F 7408001F 7E314141 4E3F7909
00AC0 - 09060E31 41512E5F 79091926 46294949 4231010F 71010F30 40404F37 08106817

00B00 - 0F702810 2F736418 04136304 08740301 61594543 400F7141 40020408 00102001
00B40 - 414F7004 02010204 00808080 80800304 00000024 5454587F 74444448 38344444
```

```
00B80 - 44483444 444F7834 54545818 0E790200 0814A4A4 A87F7404 04087004 4D704000
00BC0 - 40848D70 0F701824 4000014F 70400C74 0814087C 74040408 78344444 483CF424

00C00 - 24281814 24242CFC 78040404 08445454 54240F34 40200C30 40404C7C 1020402C
00C40 - 1C304030 4C344820 18244C10 A0A0AC74 44645C44 48063141 4000000F 70000001
00C80 - 41463808 04080018 0E380808 08000004 40088000 00000004 40088000 00091009
00CC0 - 99919BB0 000BB00B BBBBBBB0 00051001 57755550 00077007 77777771 B8F3E215

00D00 - 6420808F 32007E68 00802A06 5D0970D1 AF854EBF 0BF0BF0B F05CD206 83215249
00D40 - 42BCAC6A 2E52C223 F1111111 11111111 0AF1ACAB 44B44808 F800802F 0C4550B4
00D80 - 5C48A851 C45B0A01 A4C41A0C 5AE22A76 50D2030F 8001B264 F215EDAE 1BF5BF5A
00DC0 - F4158DBF E0E762B3 1EE0E062 C0E0616A 14EAE519 071540AF 2808015C 02031110

00E00 - E61A05A0 5A05A65A 6596D50A 4E2D9188 0A61550A C22031B5 AED1B201 E215609E
00E40 - 180A825B 1D534B9C 00C91341 5601B201 E21540AA 2A064B0A 06480580 B26B26AE
00E80 - 2AF12D91 8D290C31 0DB66B66 B66B665C EB66A145 9FBF1BF1 AE542D94 A90AC215
00EC0 - 4480D280 FF96974A 6D3130AE ABEC0E60 81C81C0E 65A66AE5 A66A61A6 6A61A683

00F00 - 18380DF8 90D00DA6 155F474B 6534244F 21A344FD 0808F152 0B044311 500C2C21
00F40 - 34AE914C 8080AE1B F5BF5460 617F3018 08F80080 3F2C64F1 30F8001B 074F215C
00F80 - 08080D21 5C003308 800803F2 C6C6C644 53018008 03F2C64E D94ECB1A 144F1560
00FC0 - 90EDA160 3301B215 60144A0E 90A5014C 85CD21A2 01E15C06 08F31E88 ED451732

01000 - 08038AEA F8EE6418 E056F346 30108080 060FD21B 476F2144 010C0CD2 80F01088
01040 - EEF311FC D0107006 1103113C 28EDDB08 E3380758 04BFD48E F631720C F829BF92
01080 - 2BF62800 005AD118 CE431CE4 808C251F 8EED3131 7F8EC14F 8E8E118E 27B05808
010C0 - C004F840 8FA8C418 F212018C 522FB1C3 94E49445 A302FF1B 344F2156 2A2E4008

01100 - 08F15421 601567D1 AE581681 615CD16F 15ED1811 54780800 38FB06A0 78108E14
01140 - 75153733 02021080 3817D2AC BB46531D 386350AC 3ABBD703 A464AF94 B3FAEBD7
01180 - B6659087 3D303102 1188FD3D E0136108 8FF10A05 40D31121 181348FC 2DE0108D
011C0 - 2AEBD733 04000184 08418E78 277701AF 8118109D B10A8E67 278AD818 723111AC

01200 - 64B0AF81 1810977D 08ACB0AF 41191089 0BA07A30 7630D60E FD8AE508 51731084
01240 - 3AF41191 08DB12AD 734FFFF7 0EF38CCF 171208C5 5D294B12 AFBBF631 808EE6C5
01280 - 1718F6CF F0133031 F995F214 71F495F2 14303D5D 9880F0DA D2314DC2 4221F995
012C0 - F220143E E4111F49 5F2143E2 50063118 E6817842 DBB064D0 DBA06550 85030790

01300 - 75085203 DE8B251C 0C98FEE0 B1132137 038FC51B 1132E013 7E9038D8 11A0134D
01340 - 9108D68F 116A0ACB A4E47094 ECD118D5 13610010 AAC1D6C9 8B2A08A2 E8B457AE
01380 - E94D9073 40400120 11A8A260 7872118E 2DDC27F5 F100DC1F 109F2C01 41ED4908
013C0 - D207A08D 956A0E28 B1D194F6 1D7D2314 DDFEB490 8B140033 38100021 081017B8

01400 - 0D8123E8 1181117A 2FD41234 00118C9E 2D81FF85 F214377D EC9DA1FF 85F22E14
01440 - 78D59D01 1F585F22 C6CEF1FA 85F22D6F DF7FEF1F 6C6F2145 031F6C6F 21471370
01480 - 11FF85F2 6D0E71FF EA031361 B495F214 41360322 FD077450 14606184 0D56019B
014C0 - 6A4E5BE8 0F019F11 5C020D90 60322D1C D077D105 E01640C5 60190207 144A4E5B

01500 - E45CD51B F18F2156 480FF540 0CD280C0 C6C68091 32CA1320 31F23000 137766D4
01540 - 00136061 741431BE 95F21461 30E2E014 11318E4D D7071340 31FA18F2 143D81C4
01580 - 1431301C 41471CFA FF1577AF F1CF1537 135D903D 51371F0F 7F215171 7FAFF155
015C0 - 7AFF17F1 45174136 145136DD 174145DD 13703342 30001F99 5F2143D8 CA141174

01600 - 143131EE 134E98D9 01B11351 7F17F147 C914503D 0A0B1C3B 125C5B1E 3FF311D8
01640 - F806315C 11F82300 6DB5D21F 4D6F215D 0037DEF1 F726107C 957D1017 2C281CCC
01680 - CC4908F1 27908DFD 1511F4D6 F2AF0153 41E085F1 471351C2 1533137E 2135A4C5
016C0 - DE03311D 8E6CAE50 01F679F2 15F01E4D 6F15B001 78DF4908 DA9001B0 44B09866

01700 - 015907E5 F8C37CE7 1BF4908D 4A001A0C 44E5DD7A 9F4908D3 60011550 6CCF748F
01740 - 4908D950 01AE063B F1B085F2 1461348F 674A1D22 28E4BC31 FC45F215 751E084F
```

```
01780 - 7ED04611 E045F157 51E0E4F7 7C031D08 E3451136 1B085F21 42132164 1328A283
017C0 - 1448FDE5 511F085F 22B7A6C1 34D27ECA 4B213613 4D018215 23E203D2 305DA34B

01800 - 88101341 61031F67 9F215741 E675F147 1351C5D0 A4E47B17 51533938 0F350003
01840 - 0015D517 5137135C A1368F27 1B11376B 6F31F2AF 514B1719 6800A554 808EB641
01880 - A6D55E03 200D0500 67E27673 77F28711 F873D27F B6142D8F 4F49E481 193714E1
018C0 - 9C7B6A4A 09EC5014 88507653 7D86D014 A51234B7 4F2C2C21 34142342 02028A64

01900 - 0D01B001 E214E0B8 43968508 530B14C7 646D014A 31F5AE51 8814EB61 9E480AE4
01940 - 84081681 6181D214 EAF73408 4F2C2C27 9501EE43 E14F0B84 40B86011 14A96890
01980 - 0B8540B1 4D1DFF30 91550851 8426A413 010E0290 AF014713 51533172 D901134D
019C0 - 014A161A 64D85313 2BFB8F1C 811440D0 7BBFC5C5 C1C12B8B 0D0E11FB 97002913

01A00 - 7C9136AD 01521201 3434C6A1 0CBCBCBE 71353440 1E21573A 36135159 B5C1133A
01A40 - E0B24131 27BF40D5 AF15932E A0F5702F A0F20400 665F2808 80E80490 A900A06A
01A80 - 0CA82018 01E01411 A1102162 1C292818 81E81491 A910A11B FF3E2301 15C01813
01AC0 - 0715C015 A09065F8 5465A065 CD75B085 A874AC87 66E8618E 713614FD 017114BD

01B00 - 8B6A49A1 B374F214 E9E6A918 114EA62A F3AE7862 81842340 84F2C9C9 708E6B10
01B40 - 8521B787 00867501 69297EAE 844D2310 1BF28FB8 35116630 915C084A 0B7F200B
01B80 - 1B574F20 B15630B2 0031B574 F20B1543 1B874F26 1EF091B8 74F21543 0334755F
01BC0 - 2D5D21BE 74F214EC 6C680D0F 6FAC9134 3C800010 00200042 0A85031F C2300601

01C00 - 01F495F2 14713514 BAE6A664 00137067 B1007135 17152E1B 4E6F2142 C401DA25
01C40 - 79987800 257A5803 7F3F3427 C10061B4 76F21468 AA600603 07843201 BB74F215
01C80 - 6070A76C 1A72900A C2B6A310 69EE6F16 2148D879 E3310214 A96CE014 C181A6D5
01CC0 - EE841303 647172F3 85885387 6A060CB7 0BE87944 866F3853 1A649F14 A96C821A

01D00 - 374F14E7 F93B6AA6 C31069E2 50AE0181 6DABB645 60617B7B 377B6B76 6E1B374F
01D40 - 214E168D 014AE4D8 A6131F59 E1B43408 4F2C9C91 3414E96A 53703684 1D21A649
01D80 - F14E79DD 7D10701E 87AF977E D8E329E5 0045E6AD D8C679E7 0ED6E106 33D71CD7
01DC0 - 6EF79DD8 55853879 368662ED 21B849F2 14E96A05 1B244F21 52090C04 7176DAB6

01E00 - 45C0707D 7223037F 4D728A7F 8F728D87 A48795D8 E598E500 45E84967 7C3011BB
01E40 - 74F215C0 6F2D7395 D0A7EA04 6FB1EDF8 00610078 52968BD1 AA49F30F 15C08418
01E80 - 67077134 456137DF 131EE14B 96C411C1 14B968F4 173E6E61 37134137 1818EC47
01EC0 - FDB135AE 41488662 175EC152 00E0E150 084085B7 65484369 8CAE87A9 15CC8576

01F00 - 9507CD41 53FFE4EE F2505034 A50444A0 849A0A45 D0B40D0E 3BA0C3DB 0541F005
01F40 - 001F4BF0 52E3130E 30406500 03008416 DDE84752 F1BC74F2 017C4179 1114E96A
01F80 - 9D84075B 36FCF713 17EF014E 96AEE840 7A935CE4 1E741185 0D07FE37 3835BF4A
01FC0 - C7DF0850 4CB73F0D 0749F158 34268468 431BB74F 2D215C06 1B885685 358E72C0

02000 - 75B2D213 04E014A9 68603102 18114C13 68B38113 65EE8478 46855858 76806C1F
02040 - 7E708407 E6255085 0137DF13 41351C1E E8E7D27A E2870E01 4E96A603 10214DAE
02080 - B13564DE 76303026 DAD7900C 2C213403 AF034084 F21BE74F 214A0379 DA740069
020C0 - BA865348 4531111B 084F2841 AD015051 6BA6E55F 87822D87 7BFDC310 2161A6D4

02100 - 0014C53F 86800848 755ED215 C31A045F E652B1FC 74F20171 6A8538E1 67F8F851
02140 - 317A3A8E 675E5F27 65C87DCE 8FD75218 630E3213 61B344F2 15C2DAF4 670A76E0
02180 - B0654F1B 444F2749 F7B42F27 10F9C300 37400A77 0005CC8E A3FE7D6F 14BA6C4C
021C0 - 07DA08ED C6F8FAB2 515114ED D074908E A0FE694F 8E00FE7C 4031F59E 97114F96

02200 - AF0B6478 608E886F 8FAB2515 CC43D7D1 031F59E9 2B14F96A AA171B64 B6555E73
02240 - EE14B1C8 14FD1AE8 A6134084 F2C9C913 5171031F A49F214F 01703176 298416AF
02280 - 87E09859 6FE8B125 D0A0FF31 C373217F DF1FD49F 2D214D1F C8220685 9AF870ED
022C0 - B6470DD1 34D73030 E0281681 C81C3475 5F2E2136 15241362 4A4E4A0A 440D54F1

02300 - 328BE531 320D5411 36180152 41362450 E14A968E 0161A445 AD181132 0C200184
02340 - B1BE74F2 14ED71AE 74F14A78 5031F59E 68114870 5815A00E 0690C8D0 3DB14C87
```

```
02380 - B0018114 A7520853 31F5AE51 93714EB6 19E0001B C74F2148 0386070A 6C01B640
023C0 - 10631B17 178076E6 80716113 66640161 14A07136 0614E96A 1E164966 2F182233

02400 - 10015630 B86B6031 FF0B132C A07DE132 20060380 D0070D4B 08098098 09136066
02440 - DBF1B874 F2AF2221 54716F0C 56F1A174 F3300511 5C3031B2 44F21524 94C008D6
02480 - 7FE01F48 5208E304 FAF0100B 44101303 DA810E69 8AF1D2CE 77001183 108BB680
024C0 - 1B265CF0 17BA0111 13134000 40CACA10 1CED114B A68540E5 171CE5FE 3174DA11

02500 - 8AC63133 CD969903 15EB0AC6 10879503 102DA7F4 03178791 0B375CF1 18A4E108
02540 - 5CE11160 5FD5F1D4 D2288018 0F2BF3A2 D59FD8B6 45AE2032 F0080103 8E0C6F8C
02580 - 813FB154 B1C325F4 D4024554 354502FF 8E9080AC 07024D01 F188F214 11358508
025C0 - E1232067 5F307136 061361BB 88F21440 71341370 61FFC6F2 1471E198 F1450713

02600 - 5011BB78 F2144314 D8C0CED0 776EF5E1 8EB21F1B 6C6F2144 726C8E6E F44FAC28
02640 - EB5D38EA EDE8E12E E0884D8E 40A27292 85079735 02793331 0F966F07 96545406
02680 - D2026DE0 8551817D 667F2014 B31D0962 6067B07A 3553D8E1 EDE70F28 588F26FF
026C0 - 062D177D E8488493 12296281 307962D0 70935C06 C8385985 88E01815 FEAF814B

02700 - 86891312 28695030 79662417 114B31A3 96633316 F8E70111 71AF48E9 B518E329
02740 - 031D0962 6065E031 0F64017A 131817E8 17F54873 32860F05 92870618 73F17C86
02780 - 46066606 3B774849 4A918726 065B3866 B031FD96 20F31FE9 66907D55 5607C25D
027C0 - 91361841 42136C28 40848849 84A8E998 05808C11 807C345F 800005E2 866E9846

02800 - 181314F7 9B573A53 15F71D48 E47C046C 73A38473 104966C0 314F8575 F1311296
02840 - 66078D53 10F5B096 2606D267 965867B1 875E0768 07753640 F705E64E E8E51CE7
02880 - 16306500 865F08FF 5FF08C76 AD8ECEBE D8321088 FB791146 06694137 061371BF
028C0 - 85F21428 F861B18E 18BE1B08 5F21468E 3FEE0713 48CC4B48 587AAC84 987B7131

02900 - C7966917 4907ED16 ADE78807 6D153F13 61341088 78E276B0 47218175 C3798231
02940 - 12966018 5571647A B45CB181 7E114D27 D2085831 5C7A738E 33C04517 3015E070
02980 - 207EE063 6E11B134 1F198F21 47135018 5A875001 37061FBC 6F2D215D 30713503
029C0 - ACC07DF25 00A4C137 1F098F21 51413501 8E00F131 F096600B F4F4037A 00500854

02A00 - 6A248498 E9DE175D F48170D2 AEED6D07 2F218066 D2879003 1E071B27 73785A8E
02A40 - B5415168 77A28CB9 217EAF50 07AE5858 8E379041 1878808E 92C06DF6 0374F678
02A80 - A15F2200 0310F962 00310496 20031129 62000118 1027FCF4 4C70710E 23D1ED2D
02AC0 - 007CA68E 1DB08EFF B051A489 7BE087B2 131C7966 028596A4 68EB2A08 578ECD41

02B00 - 47083160 6D538588 73A08701 F8487256 70456926 7EF03E11 04E02000 76365838
02B40 - 65D08587 C706F206 0238659F 8588E5F7 0560689E 8E22B07C 375D18E2 2B077A0C
02B80 - D810DD31 0ED90000 6A928789 F776E46C 878F1710 74FE65C5 8EEF3131 459663D1
02BC0 - 71038508 C27D1133 1FBC6F2D 215F3131 CE011371 F3E6F215 74A46581 314D8EAC

02C00 - 8D1FB78F 21470313 5021371F 088F2157 41350178 45071097 49F10A13 60611913
02C40 - 42015E5A F531FE96 D8271704 E0161071 36060307 76051128 E7DD16DB F2596181
02C80 - 20313B25 961B0E69 61402113 68091361 62914C91 82071360 68EF6720 7136068C
02CC0 - D37F31FE 96200313 B96200E6 96200011 71580171 AEE16179 40181148 16103DE1

02D00 - 63753018 31583163 03AFA165 7F101851 58516503 1607C001 8015C016 0031368B
02D40 - 77013601 8C307678 DE9EA1D3 B1060110 FE10C4B1 0005F378 BE3D7464 D246005C
02D80 - 275AEFE1 0C30363B 1024726F E10D3E16 FE10C151 60063901 361B4D6F 21441341
02DC0 - 610184A7 D1F13610 81B4D6F2 14286AB0 1A178F14 0E292E41 CECE1301 4C118134

02E00 - 011A1400 67417263 14B31129 66001713 1CF71CE6 0271AB40 066D01A0 5006CC08
02E40 - 74607A23 723C8444 C01A1500 6FA01A25 0065A01A 35006B90 1AF40086 9E084A58
02E80 - 01AE4006 0801AD40 066701AC 4006C608 547E1D87 35084470 C2183132 1B178F21
02EC0 - 467CEE13 0773B471 87770874 0270AB5F 060AA1AB 4004C01A E4007A72 85868104

02F00 - F91AA300 87460716 28488447 DE058520 86AD27E1 11348AA8 79087CAC 5E287A1C
02F40 - 315C70AD 78566D98 709C5812 8D08ED84 68CB14D1 AB400061 3E028AEE 01361451
```

```
02F80 - 74141878 721B4D6F 2146858B 46B465E0 13416313 684810BA C0771A1F 198F2147
02FC0 - 1C51438E D3D111BC ECE85084 18428438 4B8E46A1 8E7D5F86 8607DE48 65808C76

03000 - 7F8CE67F 1331F098 F215741C E147A4E5 008AAB01 74143134 131021F0 98F21574
03040 - A4E664F1 13D6BF4B F4BF4BF4 BF413101 778C1C12 03102171 14B9627F 0206016C
03080 - AD7D8B94 A0064ED6 1FD739B5 EBB60031 C79669D7 7DF865DD 76871361 3410B857
030C0 - 76D0D813 2130785C BF0E4318 2965F210 2313F7CF B7E1211A B064B910 A70D54BE

03100 - 7D75785F 112CCD6B F4132154 013014B3 1D396213 113132D2 14C13003 84773E37
03140 - 48A73503 0F7ADB31 D3966E33 11F768B7 71187062 86361868 611FFC6F 21431310
03180 - 3868EE11 A1358546 5AC85464 9CD8848F 5BB531D2 96150858 7A857CAE 8670031B
031C0 - 59660087 8DC312F7 21B70F98 E1A4148B 312FDA76 7E31D596 62A637E8 59D07BEA

03200 - 7AD48585 801815D2 761A2F52 01F80000 52118170 94312F96 22267FB7 B917B105
03240 - 70878628 79D37184 555749A5 3E7C1040 08707083 1F011A13 585461FB 03137135
03280 - 10A6BF58 63C15ED7 B7250E86 2607BC37 7004BE6B CE79197F 14500723 A028487C
032C0 - 42450858 86221799 386890CD 90D7779C F49D868F B31B5966 6B312FDA 7D9331D5

03300 - 966F4848 54D7FF18 6200615B 7E098FE2 C007B4E7 0EF7B7F5 5444F71F 81E11CFE
03340 - 10B580CF E1092FFB 0068DA7A 0D313296 6000171C 83E9F84E 900006CF D79DF4F3
03380 - 1718C9E6 F85874DC 31A39665 01717A88 1D800005 81878E07 11076E27 6C203854
033C0 - 6A6A7C9C 3182966E E1710184 8797F400 72B276CE 44187832 312F9663 261F8878

03400 - 51779231 2F962CE0 7606D848 034F6743 07F3C31D 39667331 1FDA7E52 313F966B
03440 - 474528EB D7F6F63F 005EB75B 05B58726 503695F1 37AC2A4E 8EA2F213 570228E3
03480 - A7F4F900 0060A985 67267471 8E801F86 B817E208 C054F7B5 9655060B 98E42FEC
034C0 - 72EFF083 02233072 E200085A A4C4808E 9E4F8591 01314F75 D87CB811 185A0284

03500 - 98E605F5 0066B571 7386B0A7 32286100 017C4C13 61341331 31BF0BF0 BF0BF0BF
03540 - 0DA10301 8E239655 017175D1 5BF8E06C 118175C1 727114B3 1C29625E 31D0962C
03580 - D3122962 0E307962 8D709155 D793E490 6E608488 497A7056 0747170D 21FAFF2F
035C0 - 5FF7FE20 008EAA4F 4E17B9B7 4A2870D3 7DE04CCB 069624C6 28B879E2 78CD7770

03600 - 7F726BDF 72AC52E7 C6244F86 0FE87800 6090656E 7D52DF80 0005BB85 98487B32
03640 - 863617EF 91348E5B 3F4317A2 2312F962 006CAD63 A172207F 50D15A07 220D1E5E
03680 - 53192962 008C908F 17159017 17C8076D 18639087 08003769 98548CF7 7F79B187
036C0 - 34001870 9E52E72B 120311F9 62000172 89312296 2A030796 6001717F 00500854

03700 - 1C18C687 FD831D07 61014B17 1960D096 2005BE17 16FCC14B 31D0966F E075DE77
03740 - EF8E98E0 500BF4BF 44CE1837 ECF7B5F7 B4B5D370 5F31A396 2F0312F9 667231B3
03780 - 6D327F64 54145178 2F781B46 06A8E791 F60D976D 08605F65 0F748072 E1446759
037C0 - B5E77BB0 86BF3862 04E4766F 7B88D131 C2966D08 7102E575 78854768 E844D917

03800 - 18E125F5 C08C336F 7A2F729A 44A726E3 18FDA773 88E4E5F4 A0310F74 81246BB4
03840 - 72EE7B30 8F670B14 9073DE5D B31C0966 907B265D ACE966C9 76CF5F97 C9377AC6
03880 - 9578C6B0 18CF93F6 AFB8759F 680F7753 42F773E5 B275DF87 BE07AEE7 1E06B108
038C0 - 611F8628 0BF4BF47 B6E70D04 C5849757 A4908597 B3E8E63F 0768FD84 471371FB

03900 - 49F215B0 13590C15 87005879 F4831063 11E7790D 474AD4CA 7D4D318F 72807476
03940 - D63394E4 8A697312 F1737963 5006D516 D4D605D8 73708790 F310E6DA F6741182
03980 - 15231623 27A19320 9B269328 857D7065 58D72437 8DE31829 628079ED 02313F8C
039C0 - 823F0373 DC730D5A 278CC75D 85F57A78 8632175C 85F47A68 87364657 88C164F6

03A00 - 5BA70F14 1368827C FA5CE862 CA727C7F 5C03716E 9B3101CE 008F09F0 0676D8EF
03A40 - 36F183E4 266DA27F 147F6242 6707C5BD 753C7648 50D49176 45315496 68417175
03A80 - 7146B6DF 88597761 5606AE07 A2250031 8F962A13 10D96260 60B08698 08C744F0
03AC0 - 385465C9 7DBD8F6F E007ACC6 1F077784 1E7AAB31 3F9667D7 63C729D3 8BCE007F

03B00 - 9C7FB150 04FA77E0 4D685955 085875A1 57286739 878B0313 F9625E86 9B0310D9
03B40 - 62607910 87807783 D3F6CF00 87932666 F31FC962 356E4F7A 2357F877 606E6162
```

```
03B80 - 2C859858 60108588 59716047 E78215F1 8787C31F C9621187 99B8774B 50103879
03BC0 - BF770B57 B7A007CF A5CA8598 E90D031F 0966F031 1FAEA8C6 21F661C8 E18EE400

03C00 - 56470FF4 637BBF53 28CE62F8 CB44F786 C5E71000 757B8E2D DE7D4C67 007C7A31
03C40 - 04962CC6 95B78AF4 7F849707 F869B063 6F865DC0 37C1CFE1 05115D00 5727A0C8
03C80 - F34DFE10 63A3D007 E0B8E40A F743A45F 600B71EB 0EA1D1E5 1DFE1062 C0D00653
03CC0 - F79FC73D 1AE85D38 77EC14B3 3E2A78E4 F80AEC57 1314C7DC CAEA80DF 8CC27131

03D00 - 049614E5 A087700A F031A396 1017D407 A6040673 09786077 BF171705 B2D0501D
03D40 - 4300D640 FE10E3D3 0007A4A7 230108F6 74A14708 31128E5D 2F027F9E 3182966C
03D80 - 071A98CC 26F038E1 07D8DD23 217BEA0E 42C1EF1C 0065FE85 931ED780 F8585218
03DC0 - 79838773 38481818 79028E0D 4F432878 617ABE85 98E81CE4 607D8A67 4E6EBC8E

03E00 - 00CE5DE6 01079A13 1D4962B1 85468AC8 587B407D CD590637 91717D8E 5F1878CB
03E40 - AF635FE1 0E2976CA 868CC54A 87841782 AFE10E28 BF005EA0 372AD8EB 8E08F3C0
03E80 - 114E01AF 3008CB70 F8CC6EE8 4A0284A7 8D985786 3EE7DBE1 3487AB47 EC93F020
03EC0 - 8FB10FC6 100D110F E10E2800 00512AE8 14B31A39 62C07970 AE450011 B13470AD

03F00 - 84A701D8 472F307A C7AC5208 EBF83AFA D5704050 08614003 81481417 1A4F4C07
03F40 - 3204CEAE 4AC9B4BA 46A4E400 81481496 40054F14 B8FAA251 84184220 8F670B14
03F80 - 70851027 64659F33 A2F27F36 5D031029 62D00331 C2962508 52037F5C 15378E65
03FC0 - 83A760E7 281E0E72 B7A0368E 98478F67 4A184082 12031C0A FA741610 93782782

04000 - 818278F8 90B155E7 EB55F033 069677B5 4C08E517 F6791722 649079E7 50F31D29
04040 - 66C08E4F 6F6D148E 9E125318 ECA6F4D6 7DB76304 33C7D775 655C5313 B962C5E6
04080 - 966E0277 DE5D9060 33308F87 C355F233 A64B237D 255F2203 1FE96651 8E4B0F11
040C0 - 18E28906 E2F63449 6A607157 277205CD CDD4BF1B F1BF1862 C4318086 150309AF

04100 - 4AEA2776 D48E75FE D27D8431 9287B739 66617A54 171AF98F 23DE05F0 682785B7
04140 - B94DE136 14A134F0 68E1D196 22231C29 666D8716 58E8FEE3 19296674 B05B0517
04180 - 1D430879 14AF9283 1C029915 52AFC20B 048E695F AF98E77B E7687850 85302615
041C0 - 3AE07244 34080827 C165A031 207EA331 08966606 A0E71B33 31828AEC 7BD34227

04200 - 914AE431 0870B35E A7806AE2 70735BA3 178AEC96 1DD34386 8379B531 307E4333
04240 - 38687F83 5C671533 14096541 3178757D AEA76B55 607E1331 78962043 02962833
04280 - 4A8A8575 65421D9F 6F6D88E8 811D4315 077E231A 8966D07A 855606C3 D34B8B86
042C0 - 7B255A03 1607DB23 1B8962FD 34C8D877 C0531707 1A285333 C8D87FD2 5CB31929

04300 - 626066C0 74923182 96571700 58636062 AE654168 E178A47A 44784285 BBB09A07
04340 - EAF47852 71D4AFC3 2D72966A 28617092 15CAF628 31C02991 65082120 8E0C3F5B
04380 - 330C9621 F313B962 42768278 E187BA03 1AA7E2C8 7160641E 67B0257E B2F5F5D4
043C0 - 74B276B1 51D71D13 11F96655 AFC3380A 072F1AFC 5C231829 65537681 3D000080

04400 - 0A1CB0A7 2FAFA7FC 17B81736 37561D97 96163446 7E031D59 665F31A0 965CE7B3
04440 - 3AA9317A 228E3DF0 7A217841 762131D0 96550821 AE0B0478 9131B596 695DB136
04480 - 16314A13 431D0962 2130C966 B183150B 04B04DB1 36163148 13434AF1 C0F67CC0
044C0 - 2330A203 2A00DA63 2B349898 17213319 8962AE34 A8A817EF 25918433 1A896690

04500 - 783349C6 3FD618D8 5320D606 1331F995 F2147131 DF13614A 163AE814 A1343170
04540 - 961D0301 965D1843 30E962E1 30C962A1 30D96221 8508538E EFBE79A2 07DA0370
04580 - 00E7E7E7 E703DEDF 13615831 36DFDE03 304D1A85 DBC9136A FC1527AF C13603BF
045C0 - 5BF5BF5B F5965000 13314A58 DC70B123 D2809E31 368BF71D F1361501 136DF136

04600 - 20038C74 E470DF6C 238F670B 14008CDC 6E1361BA F6F21564 134BF4BE 421B0420
04640 - A4EA4E41 5A4E4656 14131119 628CCE96 2B53321D 17A6F401 771022D4 8CCAD033
04680 - 20C07F4F 498D6B8E 80D063EF 31037539 50132D10 932DE7FB F3310002 15141361
046C0 - BBF6F215 64134A4E A4E471A4 E4A064B0 632F3107 46031C67 2E833200 022108DB

04700 - D5D28FCD BC007137 10916913 68B54C1B 495F215A 91441351 C9159916 4D91448E
04740 - 79DC1188 E67C48E3 5DC8F674 A11F495F 21431301 8915A915 99119135 684A7710
```

```
04780 - 40187350 A46A4640 0076B7DB 25400AF9 10980D2B F6BF6BF6 13614A13 6BB09A40
047C0 - 0FE809FA 20136156 41360384 13180961 00309965 9A851031 097AAD24 11990552

04800 - 20AEC7EC DAEC400A EC760EAE C756D716 D002002D 01361BB4 9F215801 34018670
04840 - 032A8293 600715D3 3C0D0AEC 7B7DAE40 1D27C3D7 C2F491AC 5A46A46A 464E07AB
04880 - 06548698 C119AC91 0975A086 BCE119AF 586150A4 6A4658D8 E98EED97 1DC8E3A7
048C0 - E17131C2 96660699 F8E97914 DA74AC74 FE6E5A84 1842D0DB D58EC67E 7A043058

04900 - 167916D9 D7D1CDE5 791659F8 61128620 194AD031 F0AEA01E 5C5137E9 1358E327
04940 - E8418428 4384BD0D 171B314B 31D0966D 0310FAEA 68703312 E5716C5A 131FD0E6
04980 - 63314A57 C4CAE246 DAE8A38A 383411EF 1C913615 23134A2C 5A084017 103A2C49
049C0 - 6870D192 893DB108 74431180 70484284 184003A8 680D0360 00842120 0E2E1711

04A00 - 4B3232D9 62EE33C3 F371CB5F C31A8AEA 038CF564 137D532C FB8F1C81 158E8A83
04A40 - E1CAD413 31361BBB 8F214413 0DB1087C A1D2AEBD 5A65A313 2111ED69 10694107
04A80 - 13418514 A968EE20 16A16514 61360618 D146A06F 61324F0C 2E0130D0 1523CA13
04AC0 - 07B7148B 07AE8134 18314EB6 A163146C 2C4C4C4C 2185136D 716215E5 132C2132

04B00 - 80D59693 2DF13489 FA7860A0 0B80F00B 2031FE63 8006035B 9D2306C3 1C314317
04B40 - 3B245B28 E715EDBD 579B01B2 7C4070E0 D95BCD7D 08406780 71114C4A F434EB99
04B80 - 1D731C76 E5020870 311C114B 17179517 14184031 3B14AAEC 969C3BB0 F0AE4682
04BC0 - 01C37DB0 AE0432AF 485B8628 134DDAB0 D731D7BF 0BF0AEA1 36D51BBB 8F214613

04C00 - 4118DFDD 20040315 77AF7203 F0404040 40404040 40E7781E 0E77BFE0 E7301161
04C40 - 156480DF 16015619 E7001368 09136917 DD137809 137D014A 03535552 4CD345F4
04C80 - DD1FF840 2015B531 FD0E6633 14A57739 400171AE 8BF4F48F 670B1491 171BB1F1
04CC0 - 3106AE5A 88F4BB47 52040117 1F1BF130 DAE58EA7 3E318296 65085203 31429660

04D00 - 08510313 71FFC6F2 14513501 05822AF0 AC32F304 AC530D20 7BE1AB3A 3FB377CE
04D40 - 158F31E2 96634171 14B87260 73C17DC1 5BF87182 31E2AEA0 2B46AE07 E6114B5A
04D80 - 130C816B 376E0094 A0EBF0BB 4AD8ABBA B5A3DACD 31FD0E66 31549663 1137D713
04DC0 - 76210DB1 356880BC F8531711 4B31D296 2DE30B96 28E84184 2D078217 2315BF86

04E00 - 14C84386 2A4F4F43 200594F2 294A419A 6F0A389B 5A292852 3101AEA0 394A969A
04E40 - 646B389B 1A0B3992 D34AB095 9A2AC481 032210B3 204B04BE 0AD49352 193D01F4
04E80 - BB4B24B2 4B24BF40 3BD1A4D9 4D7F822A F0AD4BF4 A3E59F77 30AD8321 05AB5321
04EC0 - D1412321 1D958713 211104BF 4B269287 FBF0ABA0 3A345009 3C70832D 0B54500B

04F00 - 54023400 0100EF28 AE7E0385 1A3F1713 10314B96 2DE01319 39E60031 039E2009
04F40 - 4A118518 52A4EBE0 BF0B8417 114B0307 8ED96D58 08E12428 417AE07B 911361B1
04F80 - C6F21441 331B4D6F 21401311 34732278 D6848849 316F966F 21767C27 72917427
04FC0 - 782714B3 10F96660 6D831361 716CAFD8 314B9E66 06482752 45606053 7A705B33

05000 - 98575F42 54429721 414BF0F0 7E56100F 4F42E79F 01102C70 F08C2C70 80DF76E3
05040 - 7D711C91 47133C20 61111311 5B5AF601 8F674A11 BF85F214 61340117 131FE962
05080 - B1313B96 22134F95 E1D1AE86 2101438E F8034001 73133101 10ADAD23 0EEA1311
050C0 - 47F6CA13 111A8EFC 03D5D015 B2172147 133C210A 133C0131 15741701 537AC511

05100 - A135AC90 331F0966 00176313 37010173 03AF8353 B1F1D8C4 074AE670 45B46B46
05140 - BF1BF1AE 50117114 F0B877B0 0B72855C E0B03312 F9660017 185531B5 7F753192
05180 - 8656031D 57355AEA 03859782 014B31D7 9EA007DC E4D280DF 72721191 35879400
051C0 - 33102181 14A9627F 1616C051 C10231C7 9660011B 10A8E533 E7481153 3D88E64E

05200 - D13411A1 0B32F1F9 35000117 43364E47 7F572517 A3F45017 117014B3 13F96602
05240 - 7F747431 31929629 077845EE 8E18AD7A 8056114B 8C764E7A 6F56A79A 67774504
05280 - 859721F8 457EE086 6A031827 DDE76CE7 94072345 FD71D072 44746472 307C5454
052C0 - 47D5679C E7395E0F 52F0A52C FC600065 FC8597CB E5FD7211 500074D0 70FF75AE

05300 - 14B314F9 660371BE 17314B31 5F966A08 610C5931 C331047E A3871925 6A78D05F
05340 - 17C7E331 2027BC41 718ECF1E 18117117 11338EA7 8D460060 1D68C2B8 117114B8
```

```
05380 - 46AC231D 79660187 80085617 114B31D2 966A0314 2798D330 6968EE12 F5E08EF0
053C0 - 2F4005D0 3103A867 36D7C5D8 6890AE27 75D80FF0 DA9C7630 D4033354 02791461

05400 - 7014B310 F9620030 49620031 CF962000 12FA9A13 61348098 B7D01501 13420038
05440 - CC00496C B4171554 31DF9669 185979F0 75AC6B00 31C27752 787E70FD 4FEE6962
05480 - A1307962 91313296 2BD7B845 8D31B34D C700D742 2AEA53C3 18F962F0 35FE1063
054C0 - 9760D663 E76A071C C14B3138 966A031A 26831707 2661E7C5 05F08656 07CD1712

05500 - 47D8C582 85731ED9 66607E32 794D4A08 77F054C7 EA155084 714B71DB 5FDCE966
05540 - 907CB151 D662F845 313E966B 07C3C14B 03E69660 0859753C 85531327 081027CC
05580 - F618D319 E9665085 88597D0C 8789014B 6B5D6EEE 1873F54E 44402355 52402706
055C0 - E1746B3D 1873F54E 44402445 464025FD 1873F445 46402B45 49502782 E75237BE

05600 - C74D0791 331B3962 A030A966 CA79A055 A33D4027 CE1AF615 B535FE10 E2972A07
05640 - 6117A7B8 597B4B8E 3A21796B 1737D8C7 6F06C0E8 597BE014 B313F962 A017114B
05680 - 037D3014 B5A114B3 18F96680 1715937A 30DA3132 96600774 01718795 D7BAB571
056C0 - 4BC31828 CC16D318 276206CE 517131C2 65E53172 6DD531A3 65D531D3 17177C55

05700 - E1747F31 4C966117 1DF743A5 CC747A63 0276CB31 3F962E07 6207A8A7 1BB85973
05740 - 6A5B0318 F9622F7A 9B629D17 18487001 4CC20FC5 400DC301 D7302D23 0FED102E
05780 - 34000878 00679179 B985856C AF635FE1 0E397600 704F8497 6AD31829 66001716
057C0 - 11F8E4C5 E204458F 674A1831 007E0F3F 56874756 27E616C6 7D2C0713 61B4D6F2

05800 - 142130D0 14A136C2 1356CEA1 818C1E4D 8C17B386 8607DAE1 717FD885 87FAA772
05840 - A44E711F 6DEF7410 0EAAA1E5 AA00631C 15B58CD7 BC17479F D5027AA0 7CE95E07
05880 - 7FD771E6 9EF764E1 7114B7D7 B5C13502 12028E86 4D8E1ACD 18117317 454377AD
058C0 - 5D07B507 6ED47F14 B312F966 61370294 E4022779 3B786E61 1A768A86 6417ECD5

05900 - D0181314 579B362F 935FE105 19761F6D D9136134 10884B84 A86B5085 A84B8587
05940 - 93A5AE85 B1711371 35109102 33182875 56452333 8D87C177 A9779871 1A764711
05980 - 913569AF 3308D875 2656063B 03308D8D E80D03D0 FFCCBA97 7333320A 8AD67BC6
059C0 - 79477837 11231A19 06B11111 33157013 3E41017A D6112149 3308D813 710A137D

05A00 - E80D03D1 FFFFCAA9 9777720A 8AD67476 7CD6863E 111A1360 671E6071 36135319
05A40 - 27F76683 F8F670B1 401F0F08 E2A2D602 F31D19E6 6069A37B C75F1F0F 08E082D1
05A80 - 4B7C0517 19663F6F 9E6FC133 82927A7D 5DE76157 6F415341 70A4680D FA99A42A
05AC0 - 42A4CA4C 4D03582C 29258033 829280DF 0C7B5567 4E14B119 E6109136 134135A0

05B00 - C1C114F9 6E7FA0C4 C031C214 D57E31D5 72A17EE5 31B514DA E06DBE84 B5508537
05B40 - A7486351 75547154 3364E423 A95A46A4 EAF41021 4B170137 10920A0C 45213613
05B80 - 40618114 E96E7FA0 C49231C2 14C57EAE 07DE3798 411980DF 1127F686 6CD11980
05BC0 - DF07161D 5136E113 61348905 08091378 68501751 711377BB 3061378F 4F0B11C1

05C00 - 318214D7 72411994 AD080DF1 1A154107 13420799 079E4605 D8E464F5 F0790417
05C40 - 3137109A C5119135 E6109665 0AC13025 2533A64B 7B434F58 488437BE B3B6BFD7
05C80 - 6BEC77BE 7A06EA7A CFAA20EE 7A0E4B73 1008E2E3 F1C114F1 02129AC9 129A8A66
05CC0 - BE31928C 020D311E 9625085A 848DB135 10A11013 6134E28E EF7587AA 0318275C

05D00 - F17115B3 171137D7 2096C433 1C2AEA70 7211A8A3 A0DF1355 4D87A607 78F11913
05D40 - 51C114B0 37C5044D 31A79EA7 C330FFF7 45249031 07B6A337 828966A1 239125A3
05D80 - 0F912D92 031B25D8 31A89666 0E7E77F3 47C926C7 F7094500 DF135D61 4816114B
05DC0 - 17114816 19661F13 7D7028E0 C2F4E1D2 72EEAEA8 0DF8E536 F1C48C15 2F794210

05E00 - 2173D014 B137C2D0 6F5DD27C AEDA80D0 137FE0C0 C809FA13 5157717F AF580FF2
05E40 - 1A0C5223 03A8A208 11A84793 1B4652F6 ECA6FA01 53320320 059B6E58 40321109
05E80 - B61E3103 AEE811A8 47DF005A 6E04581B 46A4E462 31E2761E DABEAB46 52DD5A65
05EC0 - AC1480A4 E41C1726 E5A85005 FC0480DF D280F0B8 AC2AB532 2109B1D1 05A4EA6C

05F00 - 480BF553 F3103AEE 49881131 03A0975A D31E27D9 DAEAB464 F0811A84 76505FE1
05F40 - 433354B2 860E0B26 B2605F80 47DB880D 080F280F 0D520310 3AEA8227 C0078007
```

```
05F80 - 400684FA 84BB5832 008C75DC AE514B17 1B46BF1B F1038BB0 08CA9438 CE16EAF1
05FC0 - AF274DF3 1D296690 30471CF3 3069676D F4D03103 A8679AFA E8032F30 32031467

06000 - 61031A07 E00A8881 5BF5AC50 3BF59E2B 0B6AB455 4FBF5AC5 03868003 58564E42
06040 - 58C0E3F8 5814373A FBF4BF47 0AF31038 E955E2F3 0B491BF5 BF530996 5B0BF5BF
06080 - 530720AF 40313213 0131D584 31C114B7 A915D01C 114B9667 FD9700F4 5085396C
060C0 - 9D030617 31331611 36134135 18106D5E 877CE8F4 F0B12007 134071C1 14D1C103

06100 - 86300318 26ABF863 007CAB13 6134D513 31311711 338B8C11 3314F655 18888E21
06140 - 30F14D5D D200133E 3D323030 CB920CB9 2AE6A6CA 244CE92C 4F0332F3 69221DAE
06180 - A3232D2F 926CD6C9 0B025FDB 06453B06 4D13702F 42502270 337E4F44 50241F3B
061C0 - 025485F4 25022B03 390214E4 44022903 301902BC 302902A2 82130690 25330A98

06200 - E0930C90 8B0A86A0 6B8A80D0 3FD262E5 A2F252E2 B2210339 02449465 0241A312
06240 - 29620030 79620003 31929660 0119BF6B F6BF6136 14A136F0 F09A0000 33210057
06280 - 032F0080 101DE062 078AF4E0 07DE80D1 689E1711 4F962BE5 4F000000 00000000
062C0 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

06300 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
06340 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
06380 - 00000000 00000000 00000000 00000000 1F088F21 55401B1C 30202E41 4D454020
063C0 - 20235024 59505540 20202C45 4E402020 20244144 55402020 2024594D 4540205F

06400 - 42545D0A 0FF310D9 62606981 AF08D8C7 D1323FF5 66DF8EFC C35817A4 47C417C0
06440 - 28E869B6 2452F30E 207A4F31 8F96261A C215548E 1FE34E96 5318E79B 31F678F2
06480 - 1458E56A 01FB78F2 1451FDA3 608EE67B 1F678F21 4313014E 96AA51C4 1411317B
064C0 - 8173C08F AB2514A0 8E85CBD8 1F178F21 43DC8E9F EB335202 34402AAA F3A49055

06500 - 960B6B01 00537620 1D4D2310 2CA13014 2136CA13 014E96A3 9668FD01 B678F214
06540 - 68A5B08A C8E667F1 3416F16F 142136C2 1321345A DD413177 1085B84A 8E72048F
06580 - A8C41640 48C36BA1 BB78F214 2D859970 F64C38E8 B934606A 8E876A29 4A12ACBA
065C0 - 4E441A4E 4C172B56 08C52D36 B9E6F3E8 C9BD2B67 57171BD8 E25E3593 6D21618F

06600 - 8E61A341 F1376C6E 1F088F21 57494AED B465B18E 12E34FC1 4B9681F7 E4F6DCF1
06640 - 5548EEFD 362EF840 72F07906 8E3A5B8C E3CB4812 8F785C18 47AC3A4E 94A918E1
06680 - C838EC05 35468579 7CC5850D B10B8F67 EE011BD7 8E69838E 6F65ACBA 4E5B4D88
066C0 - EB493CD4 5713514B 96895CD4 E68EB380 59E8F156 816D317F 84704D08 319E33A3

06700 - 006CDEA4 E5C2B674 CE1028E1 093112D8 13751A8C 13D38EE2 9E6BC08D 6B7D1877
06740 - CE7D318E 4C01AFA2 F30A2F8E ACCEB465 4F8EC2DA D913415B F158F16F 1617C745
06780 - 402C8326 00D0D370 23505542 014C1637 F5117F70 211C471E 02431F21 5C916917
067C0 - 314B2070 E0F2BF23 1F215C51 651C671B 015C916B 1378E02E 34C0A4F4 60734031

06800 - FF14C860 008F1568 18E058E8 F3668113 37C34131 76008CCB 141BB98F 21461340
06840 - 1D2AEBF6 DA3303E2 A6290B52 15C3163A 8BA8A8FF ACE00478 3015C316 50114C16
06880 - 9011FD55 F2147135 01161157 330AF2F2 1C014FDA 2E331343 20D5BF2B F2AE9A86
068C0 - F4A0D5EE 03AD0143 EAE481C8 FFACE035 5949198E 74F23302 D214C012 98E72BE1

06900 - 898F3501 1423D015 B334FFF7 08BA3123 B982E181 70CF1612 E72AFD23 050375E0
06940 - BF6D0AEA BF6BF615 C9169D60 3F61DF75 228E21E0 7A25F9DA 7F0F8E2C 9A4518E6
06980 - A318FC27 A18CCB02 8C10A2D6 2DFD2310 110910B8 597FE18E 6CD01111 1B1BBC6F
069C0 - 215C3682 0E6061B8 E7F28FED FF007521 0523A1A5 40E404D8 8F6EFF04 72D98E00

06A00 - D28E856E 8E717E31 027802D4 8D721018 C3C90D41 3216115E C1300130 EEF751DF
06A40 - 31F16530 876C0703 1C06D7B6 78B7221E 26E128D4 FCA1DCDE F121DF31 F01F078F
06A80 - 214D7BFD 8E376344 2311F962 B18EDB83 5907BE14 FA8E3B43 47979045 3A79C087
06AC0 - 64A74314 F817F143 8E65C0DB 1F688F21 457C705F 08E084E7 420135D0 15B311B8

06B00 - B6111378 E0AB0135 48D667ED 91348F24 581136D5 1FF85F21 478548FF 1E718F1C
06B40 - D711181B 688F2142 8BA0C01D 8D08DDFF F0161136 13511175 EF400870 B987A690
```

```
06B80 - 38C0D838 49D2E610 93399991 0B84A14A 311F9660 0160142F 41641018 79A085AD
06BC0 - 010314A9 660084A1 6115A310 38790011 98B24001 8CF2351F D55F2143 13117F17

06C00 - 314F1C38 2281E0B8 70200B33 D300018E 4C0C1180 68E568A8 E758AD23 1EB8BA61
06C40 - 3314008E 7772D231 EBDAAF1D 881D0710 88C2D7A8 7642711F D06E5907 D7730FF2
06C80 - 4D083121 33F30060 598C9923 61FC8ABE F4EECF78 EF43E8E0 B2343B13 31311007
06CC0 - DF152B71 3F43C76B E1108660 36EC01B3 A5F21421 30166142 0106D910 B740007D

06D00 - A8CA0228 E90A0758 17DDF8E0 F838EA45 A12BD511 B8EE96A4 A98E24A0 7EF04B38
06D40 - F130011F F85F2143 131768F1 448F271B 11378ED2 7A8488F0 7FF063CF 86D808CB
06D80 - 1908EBCF 01B3A5F2 142D230C C2DE228E 1B1265EB 1318FF2D A178905F 112011BD
06DC0 - 78ED0F11 33540CA1 1872E0CE 761F11BD 58E46E17 2FED65E1 7640D5D2 17114FC2

06E00 - 8BD6874D E144135A F215F3D1 AE58E18D 178BE130 161D214E 8F471B15 8B7F9E18
06E40 - 51461318 BE000313 313186A0 0D2876C2 17314FC2 13416331 0F8E93C1 1611C313
06E80 - 6133D5E8 0317114F D5C20378 CC70BF50 0113067D AC5908F1 3001DA07 D8E1DE03
06EC0 - 84617E14 3F41CE34 412E08A2 0085633C 02E8A200 017D2013 514B96C3 F017A101

06F00 - 3514B968 00D57BEC 8328E687 DD9DAD23 102C2135 143C2019 D5EFA7EC F85B6130
06F40 - 85A66008 4A8597CC 363717DC CF85A849 60614A5E F84ECF14 A31ED966 8E161711
06F80 - D5418C41 4287B551 116DC02F 30720943 D097C808 C55E2101 7D528438 45ACBB46
06FC0 - 49CA4EA4 64B12F30 1209C3F0 86B4B943 508558E6 45A48911 18E0641A F0D38538

07000 - E554186B C10B068E 7F538EC2 1340D84D 070B8750 30B1B198 F215C28E D2211B19
07040 - 8F20A15E 20B40511 91355E18 E4041817 8E11F245 386B7094 F8B84473 8670C653
07080 - 334402E0 8A242775 60348033 F30086BC 0068E08C 00768DE8 541CF1CF 8E824354
070C0 - 17C41844 72267E66 40B72215 028E94C0 8EA9A087 B908F011 218EA1B0 8E9A8B86

07100 - 4A07F166 0D087970 86B606B8 01B678F2 1468EE0A 2C213414 A161311F 96680142
07140 - 16131F09 66831648 FEDFF040 18717086 06061751 3374554C 01301816 F308D6E4
07180 - B179A154 F87851AF 871F545E 8ED50258 174F7560 6BA07940 5B07F658 6A211360
071C0 - 68E02F10 713486D6 06513844 73718646 17305135 60699D13 60601644 286A001B

07200 - E76F2146 134CEE60 11368ED1 92E21B67 8F214403 71F08E57 7B763585 E71CF5C1
07240 - 8E94908F 01121784 75128C7A 091F765F 2143CCCC 8B6B0852 52713779 A0145851
07280 - 13514F90 EF01718E EDCD6230 1377BE58 EBBDD8EA 6ED33040 28E64AB7 86013117
072C0 - 18E7DCD8 E94428FA 8C418E2B 3A8E8B5A 8428E924 9591268E 30A91841 4A81CA04

07300 - 4BE76108 62606E33 7F001300 86BBE8C7 7FA1F5C8 F2143018 5831E096 6008E538
07340 - 2500B475 00848017 E3685D31 2C8C7719 1BE76F21 44134161 1327B431 30550181
07380 - 311C65DF 74DF77F4 8D871016 88CF315E 14A966C0 1618D1AA A1136109 1367DA31
073C0 - 1913414A 16131F09 66721648 FEDFF050 18548D80 101CED58 C906F658 D723F57F

07400 - 878C1103 702448E5 1D8E5B7B 560683E8 F1610140 B7C654DB 8D080018 4E136108
07440 - 7CD15F11 813486D4 314A1612 090C7287 D606B611 F765F214 71328B26 061D1132
07480 - 1631F976 F2136145 13416108 D014A203 15B9E291 1618E971 106019BD DFE16BF8
074C0 - F0E5F06E 021E167F 14FA6690 E606AE01 36064258 41840201 FE95F214 71C41458

07500 - 709D87CD 080E834B B8241360 68E20298 7C606990 84C71F08 F8E946A7 27158E86
07540 - DD7340ED B0D5AC01 F167F214 FF590906 B440E019 01FE8E58 B01438A8 7D744140
07580 - D100D41F 167F2159 08FD7521 75B54010 70613416 37BA5110 D6856844 8CE4A007
075C0 - 13484087 E60658E7 B0177117 3614B286 D6287112 16314A31 5F183966 808E7A41

07600 - 136718D8 70908F0E 12179006 F84E673C 8DA05218 DCAE918D FA5913CC DFAE3CF1
07640 - 4A318F96 2451F7B5 F214F96E FC8F4679 14CC73D6 8F4E0217 3B043387 2B286D50
07680 - 85E8518C 3C3177CD F129CF64 BF852788 656C84D0 8652F1F2 44F21534 9480085E
076C0 - 018C79A1 B3CDF5E8 CF85E8C1 73184D31 2C8C8ED8 8C96D220 1FD55F21 4313117F

07700 - D0880901 5B310217 F1430370 EF5118EB C4F42420 74CF1371 34C2D716 E8804112
07740 - 234412E0 8A690122 200333F3 00102027 3CF5008C B2C17FEF 8E874F6C EF7EEF83
```

```
07780 - 20052E1B 265F2146 13418454 1181316F 14A16196 2C11468A A00E6400 CE132CA1
077C0 - 3056D164 7B101849 749D1851 4AF0B244 50183037 D2021152 7B044C00 C4900C52

07800 - F0D0DAFA 15212003 8EA48DD5 3F020202 02020202 020376FE 316F8EF5 21440021
07840 - 81132D81 32168769 F11B9724 1D4130D0 14AC0130 57CD4130 18114AF0 B244B018
07880 - 11365F41 36D52176 9E161136 1348B500 D98BF00D 01368B53 2D713416 314A136C
078C0 - 213414A1 61908CD5 9EDB1351 34181038 49163146 F6D51838 799286D4 21428A8C

07900 - 1136CA13 575BD137 13413146 C021F765 F2147D71 C4147D05 F1D0D215 F38A1921
07940 - 7314B137 8B736C21 3514F171 90A9D14B 13759E13 71358BFF 3D78F291 01DB135D
07980 - 95D01321 30E21440 2311C6C4 D7B8D136 135D2311 18A20003 E7BDFD90 BF14A313
079C0 - 2966908D CA11185A 310F9E29 2D262F0F 4BDF210B F8536010 E3BDF100 BF84384A

07A00 - 84484513 61097871 7D314908 F95EF011 38631607 DA875808 EACF0AC6 87944AC2
07A40 - 864F01F3 86F21415 F287DF11 008EE97F DA8E5A41 110AC2B4 65D08668 0AC6B468
07A80 - EE8411B1 78F21504 11913414 A16131F0 96291729 8526AF87 3DC48513 65F0722E
07AC0 - 5B41C113 786AF01F 296F2145 501873F0 87DA07F9 86BEB134 7D504F08 FB7EF011

07B00 - 91347748 6ED98631 18E7241A CBACAA4C 86401AF2 1F386F21 5D986611 1B178F21
07B40 - 5A077748 E521167A 88DB1EF0 7B2D4001 731371F9 76F21450 187D0031 1C8C8168
07B80 - 72AB6E00 75EF4007 CCB136D5 1351C914 38A8E3E4 4C11C213 7C213570 30137D7D
07BC0 - 46900179 1331F265 F2DB1411 741451C4 037B4068 AF14B311 C9627017 8031C314

07C00 - BBE4B045 501731C1 031FD55F 2143D231 02CA1311 47C2D717 413710A1 34135147
07C40 - 8AE00137 8F29101D 91354808 C5B62169 174AC1D0 14A16190 C5016313 68BFB313
07C80 - 614AD88E 157A6FB4 09B640AB 1401CC20 2C920001 32C02013 218151BD 0CC14111
07CC0 - A1351410 3210C137 12A135AC 11611321 30137135 EE94D501 45131181 890CA137

07D00 - 12A135D2 CE145891 01132C01 311C65E8 AC1B4550 88F98401 1C41438A 6851F8A5
07D40 - F2147719 E7D8E798 E14528D2 1FE76F28 ECD691E7 B5F14D7F 1C302D7A F0DB8F25
07D80 - 921A0F53 F73987F0 31371341 318F3D69 1133D231 D3CA1311 848FE61A 11648FE6
07DC0 - 1A1DE8F9 84018A68 C766F8D5 979107AD F9BFBF31 1F6110E5 ADF7AFBF 317FD531

07E00 - 0EDD14A9 6041E596 0F08F806 31644421 8F3F5315 2F3C6DF9 FCAF14A3 13E966E1
07E40 - 1F386F2D 21451741 61136145 6604B069 62606080 7E527D01 1411618E CF278EA8
07E80 - A604948A 0AF2E651 28F30231 BF697AEE 27A90A1C 99E50AF6 2010874C 01471097
07EC0 - C3213214 17C12110 1597BF01 198F9A13 1667F843 84A71B1C CD6D714A 30D90231

07F00 - A0E90280 85A55085 3161CF4F 514A31F0 966B016A 14A5E3CE 96673161 14A314C9
07F40 - 66512116 114AB045 6F204511 361358FC A9A11371 34A0C908 1A674184 562F01F1
07F80 - 78F2017F 5DFAD3BF 84514A31 3E966611 F886F2D2 14587500 524E6966 A27C01D6
07FC0 - 814CEAD0 D08F2592 1722150D B23DF5DF BF87D808 CD3588EA 758646E0 88548EDF

08000 - 90DE06D6 84A85513 414A1611 3610A103 8532030D 90671843 866F07A3 0113B441
08040 - 037B0B48 08EA0E71 1A1347FF A74FA480 8E34E711 A1348496 6990B06A C6D0810C
08080 - 4C4D6706 0AF015B7 BF08F2B1 31070B03 8E0E078E 06868E8F C35708AC 008CD5E3
080C0 - 31328CDC 211617FC FD23038B 64003707 3B340E53 D1FE96F2 1336910A C6D28121

08100 - F296F213 3CAC6C6C A1310314 2761042A E6E6E616 3D472050 601D8133 1018EC56
08140 - 1584D906 15F596A9 3965E2BF 6F6F5F5A A1D4B618 EA64C074 E0175147 7E0003D5
08180 - 0617A52C 07121131 1110234F 95E1D1AE 8C9C5C5C 5C913701 675DF769 BF8566D0
081C0 - 0165DF64 9BF218EE 539445AF 33302021 0814A313 F9663116 1873C18E 53F85343

08200 - 10F9E2A0 8734158E 310D9667 1307A871 61AF05D1 8C07118E 92125B08 E0FC14AE
08240 - 81373128 73908536 68F876D0 7D004BC6 8E78CA00 28EF5F88 60C479D1 804D1831
08280 - C1A074A0 75716343 2033A300 6E616461 10111B84 8F796B50 8588F978 C1664186
082C0 - 2118EC10 98626062 41727141 C13710B1 378E319E 4FAD015B 38EA44F4 901137B1

08300 - 98538E5D F8872398 453FB656 97370202 02029765 085511B7 A4C1458E 51E85708
08340 - AB328138 ABE03019 03606F1F 7CE04C03 3B30061A 08E07F88 17770C14 71081358
```

```
08380 - E873F120 12286571 34C02E08 A2B033F3 0052C110 D23102C2 701140B1 1A134119
083C0 - 1358F261 B17A9011 91341507 7AF11861 42F43480 2E08A631 119068FF DD010710

08400 - 98C2E190 674FF070 21038538 E2CE8102 862606A6 E3019036 F8FD12D1 11A10913
08440 - 416E6A9F 8178CA2B 18C00028 13844601 08436600 8538541F E95F2147 1351C81C
08480 - F863801C 81CF874A 1151717F 11815D31 73DB1450 3153717F 15F31081 73147D70
084C0 - 3AC3108A CBB464A0 A4E94E37 8E7CD840 01301D17 147CECEE EC01318E 6FF27DD0

08500 - 77B01B76 5F214606 16414212 18E9D121 191BC65F 21441840 71448F6C A21038C0
08540 - BE88C18C 8B675C18 E2FE14DE 76105008 E8EE16FE F8EDAA14 3D8EF702 5751378E
08580 - A69EE6E6 110C24FA 068EE2A7 DA078B2E 9134181D 214C1101 36E21347 220D2310
085C0 - 2110D8EA 16914003 8C64B185 06600840 132101D2 3141CA13 0D087040 2115018F

08600 - CAA218FB D031BF6B F67E1F11 113016F1 652015C9 03342A5E 1C2C4C4C 4CA13214
08640 - 6132C201 6C946706 A5CCF8CD AF7C13D9 06582136 06173133 D23192EE 218E5D80
08680 - 07134133 100D2314 38EE0C84 4B07D511 81351C37 982D4159 38EFFE68 E32F630E
086C0 - 902587B6 2760314A 316F966A 176F205A F2287882 5C185249 1AF0B44B F41CF151

08700 - 77F22842 17F17F13 31311577 AF717F13 7DE218EA 280AFB87 241AF572 12473888
08740 - E21CF052 031418E5 253AFA13 70676910 713517FD 296A607C B18EFB31 57176821
08780 - CF1C4141 78A16951 8EF2E616 31461081 F765F214 7D713606 87DE07EE 2161136D
087C0 - 70713423 87840218 0FF1837B 2220314C 75B24A03 1A268648 0DF11016 11561916

08800 - 3D183736 44017844 8E97677A 446D3206 7897D813 3078B6C0 C08BA505 5331B265
08840 - 1484ACFE 0CAF7A21 59E1CF1C F1C474D0 1478AA4D 86D3B8E3 196AF98E 18061B17
08880 - 8F215471 AE95F146 13416415 678E2804 AFA7E401 13AF88FD 55911747 1905131C
088C0 - F1C41438 6F911307 9934217E 738EB957 11213084 16B717D3 06651048 6FF11F97

08900 - 6F214713 51731371 FB88F214 573D01C4 74C01517 8CECC617 F1731331 F995F214
08940 - 11741411 31031537 17F1577A FC2E9197 10594980 BCCBCE23 8E00B404 0185815A
08980 - 38EA4CB2 3490848D 121A9823 1FE95F21 47108174 1431358B AC117F17 F1741479
089C0 - 11D01731 3750E0D2 0011618E C9768EB1 F5047800 14513503 8FD55911 CF137038

08A00 - E2311D01 4AD8132C 013003C3 9CFD4BAF 67201621 6114EA66 56F90AF2 8C71AE41
08A40 - 9CF50BAF 8411B976 F2142130 D014A136 CA1308C3 8AE16116 31368BFD 9136D014
08A80 - AD816114 A9620013 2C013218 114A1619 0CFC59C2 0310F7E4 F14A9620 0161754F
08AC0 - 50FD1BCF 39CAF71C 214A311F 962117B4 2666F8D6 CC71AE91 F178F214 D1618E48

08B00 - 6631F096 6CD8FE83 B1171143 D8D231AF 8BA40D51 7D14AAC1 31B3962D 030A9668
08B40 - 0B45B451 B178F2D0 14ADC763 01121593 17315141 701B995F 2146134B F4BF4D23
08B80 - 05EE16F7 3B76CBED 2305C2BF 2BF2AE9A C910A7F0 1112AF8B F5BF54E3 D2868733
08BC0 - 152C1728 0D972F81 191358E1 97115DF1 7F33C02E 15D317F1 74011331 F0A8F214

08C00 - 17951143 CA100E15 01D6C913 574315D1 7720C013 1E01188F 401B18E0 CA111B8E
08C40 - 0D988D69 5C18E656 85006E37 13610A8E 5F1711A1 34018CAA 1776FF40 08C72698
08C80 - 0F020D78 EA16846C 1D99143D 6E91358F F61B1F91 36DB80F0 7FC2F903 8EEA61DD
08CC0 - D78E8E21 4D584813 3103D231 02CA1313 150C2134 143137C2 DFD51361 341358BF

08D00 - A2D014BD 217114F1 C18B8A01 33C25DD8 A00001DB D503788F 50077201 84142EA1
08D40 - 40D5F913 3131CA10 08F861B1 8C8F8106 137D7113 8EA8ED83 2606F16D B1350701
08D80 - 6B011351 53785A6C 008EEE36 84A8E79F 2848849D 23048BE9 1CC14FD7 565D187A
08DC0 - 008C5513 06D81C11 37C21351 4B313296 2F9078A5 AD8581C1 14FD78FA A2513164

08E00 - 962C0859 E69665B3 1029E7CA 31D79E33 ADBDA37C 5F506C72 78F890B1 5981F22D
08E40 - F1D1D217 114FE596 75F31838 68F19EDB 0E158F6B 5F86940C 6C1DA618 0D09ED7F
08E80 - E19EDCEE 155E07C6 8B13D81D CDCDAF0F 01C114B8 F670B147 BBE0A0D5 7EF48F2D
08EC0 - 2B18A80A D2D1DC31 8A8B1193 1838B870 CA58FEA3 04C29658 031E0E13 1C2C2961

08F00 - EAE69617 A03290D8 0FF20AC7 D606D230 6D5221B3 A5F27F4D 07ACB812 15D5037C
08F40 - 50400AF7 817D6165 132221B9 95F280F5 25A95E8F 91311421 34E28E87 5225A898
```

```
08F80 - 0F51B995 F2146C91 441640D5 2F20031B 3A5F2142 1641468B E0013015 E5033210
08FC0 - 862E7E33 CF8EFAF8 406F10D2 3CF78CAF 85078F07 18C46076 6C78AF53 2860E070

09000 - 7C31C26C 836D3A8E C06EA362 EFAFA814 102B4442 DA4CA445 C094C9D8 600D1F38
09040 - 6F21478A 670D2145 7AEE8603 B112AF6A 465F094E B1740C06 01A4C551 7FD05427
09080 - FEB31716 B0394833 3048169C 6AE8E2DF E11A8E90 A11347BB B4808E2D D685D6B1
090C0 - 07DAB757 EDB1BE76 F21447A3 08C508F8 7D0085E7 1DE13310 18F97911 111E0100

09100 - 1F495F2A C3B478CF 9518E2C5 E8E278E8 E9D0E432 8EF32E8E 357E8EE3 077E7E93
09140 - C008C265 E7F6E93C 008ECFBE 5AE1B265 F21468B2 00164146 8BE0003A 41CFBE2A
09180 - F8EFFF58 E40C2042 E5A091CC 411091C4 47A5814A 20908121 6314A181 315F9625
091C0 - 18EE38F5 DD79D88C 617F1368 E889E134 1618E4F1 94F3EFCF ADF008C3 08E1FE95

09200 - F21471C4 14501401 13610A8F D4D0011A 94AD6137 108755B3 1839E176 1107740C
09240 - DD04B2D2 308DA31E 0E148081 C57F136E 913418E1 5E00E06B 98AF2570 2E301112
09280 - 8C79F513 18DBD3B1 18D2D152 165DF642 B55E6163 7379676E 65646FFC D1CF1E4A
092C0 - F7DCA343 44F2135D 0808F153 0C2B0440 11510C21 35D914D8 080661D4 11136108

09300 - 1371097A 7A56064B A73A9D71 B495F214 6DF41211 1795F110 130341A2 906DD48D
09340 - 771B18F5 11511031 29137134 7D2F1371 351298E9 461113CC CC41114E 1618E336
09380 - 15CE1101 308408C4 B61D5D2A E9735085 E8418508 C441E80F F041088C B901D0CC
093C0 - 46028D07 2EF3F4CC 8ACF031F F8EB97E4 008FDBD5 12A54113 678BF2F8 0FF2C4A0

09400 - 11883100 32F00108 A465B289 A62AF21B 6C6F2146 133EE540 D281E782 713710B1
09440 - 10D67650 8A6E5207 B4074006 64F0480F F2C32F00 1088EFA1 E1F83100 7F7D8F84
09480 - 4108EE15 98E55794 9106AF21 5D58451A 810013E0 28FBD410 8E877858 08EFDD87
094C0 - 81286D60 7B121188 0DF7E618 52890B02 00B3080B A4655085 1A464F11 A4E7F15C

09500 - 386D0116 315A3163 15838720 3316D8EF B1E86D12 1A386F14 68AE3116 41468AA8
09540 - 08C4BAE7 1E5A464B 5118D578 541F0E3F 1862801E C0BF3047 3B218386 D921F8E7
09580 - F2147F28 AED031E7 78835111 63313371 9231A372 73291187 AF3100E0 310CC28E
095C0 - 6EA8C49E AB1D98F7 94108E4C C81188F7 94107021 863808E7 66586080 8E38C88E

09600 - E9C88EE8 2830F8F3 AD318F6A 41087202 87120751 04C286D0 07205181 8CD91E11
09640 - 880FC0B8 0F00B80F C1080171 801F4869 011B8AA1 18F43581 11B13524 D114B171
09680 - 7F9033E3 3373501B 6C6F275A 4131182A F015A281 CCC4607F 60D08E48 5811B7B8
096C0 - 41358FFD 1511C040 08F7C151 52F2031C 38CEDC88 E6576521 73448ED8 1E147540

09700 - D28CA7A6 7024131D 0CC4F08E B49B135D 4220B80F 10B11880 FA7B24D6 14B875A2
09740 - 962A2E58 76810613 71088E4E 48118135 0717152D CE51E866 81133E0E 0133D48F
09780 - 51E71118 0B80FA80 F10B74B3 10803203 2CFB8DAB 8118ED74 85908F84 F517ED11
097C0 - 31EACC13 632B20A3 A511D281 C7A31A6C 58F8F741 518EE348 5008D57F 51008E42

09800 - D977918E B389136C 5E18BC90 8D066816 D2C17081 6BF6AF5D 2E60480D 0BF3BF3B
09840 - F3AE7BB3 AE71321B 995F2AF2 146130CE CEE24AB8 1E92BE3A F0A94A27 46015312
09880 - 092BB206 A275628E B1450432 6360E63A 63BB3A87 52262A04 47A8BF2F 2A27450A
098C0 - 4EA27550 AA7814A2 E59F07D5 32934810 908F0AA3 98A70B0A F6A8692B E0A2F94A

09900 - A031027D 00A0F5AC 4B26B0FC D49F14C1 6103D5AF 67AEFBF6 BF6A0FA0 F5FEBF7B
09940 - F7BF731F F14C0320 AFDA6F45 ED281281 296B71AE 714F1719 63AC7F9F 50FE6E14
09980 - 39136137 136C67DA 91361371 3651A7B8 11F4E7F2 1472FD5A 9280CF10 8D9F6F6D
099C0 - 778DD96B 8514F96A E417A967 2F1C4143 1331C815 F38AAA11 33CA133D C1478E5D

09A00 - BADC5B01 311745CB 173137D7 1375B0D1 1FAE3F11 1894E40D 1D9D1143 AE8B644F
09A40 - DF4F4962 D0137C91 3752E173 30C14398 A67118D8 F5B044D1 1727AE01 37108B04
09A80 - 4F9DB135 58A171A0 D467BF2D 98128129 6A6B112A F8A2E426 13310292 A01747D3
09AC0 - 102D5311 8766D757 010A5389 0271CE98 270170F4 D6753D55 E11894A0 08AA4277

09B00 - 401085EC 143D84C0 723010A1 1829629E 7F0D136D 51101B49 5F26A001 B976F214
09B40 - 21300111 A13529B9 6B96B96B 96B96200 329B92B9 2B92B92B 92200320 8EC197AF
```

```
09B80 - 01008477 AD18E1F5 57293857 8E39128A 8D2D8137 135C2AF3 D731A314 B96254D2
09BC0 - 30E8BDA0 3075D003 A8981EA0 E8160710 88E823A1 1806AC35 00137135 8A7906CA

09C00 - 0AF01001 4B17131A 39622179 62500DB1 3564808E C83A1771 371358BB 50582D5A
09C40 - F63705F4 25459766 06841D9A F88A3907 D12565DB 135D337D 41494E49 71A23734
09C80 - 14254497 10133053 497562A6 F2F306AC 7B47B471 1020A4F8 77000702 07759730
09CC0 - 47083100 03136135 078EF22A 100B465B 08EF72A4 0E137134 71700650 B18135FE

09D00 - 10E32591 6E916557 814A3182 96290A6F 608F1101 F188F215 17071618 E5445736
09D40 - 117F1371 F188F215 37100135 78E106AC 36F3F15A 5AF31618 EE6680D2 2F1D59F2
09D80 - D32F2E23 FFEE6F4C A3F00181 01DB0613 713525A8 320E381F CF48514B 31829665
09DC0 - 4D177C04 6385131E 29625184 173705A3 31E29667 17370552 7C505627 5505228C

09E00 - CF8C7B70 55FD3A6F 4638718E BB1BB132 5169E18D 9A93DAF0 AE4BB5D9 D78F2D2B
09E40 - 1A86A870 7135611F 8F670B14 6ABB1A88 7A204993 19296600 7A104907 8005C803
09E80 - 14B31029 620001CF 40017114 B011618E ED258E26 A4470958 7094C662 CAF23251
09EC0 - 5238EDA5 3515E4E4 8EE182CC CC100AF9 1098EB08 2D37A100 5BE31101 19AF58EB

09F00 - 082B34B3 4AD48E89 E131F09E A808CCFF 1A86A878 CC965754 C2033020 21085018
09F40 - 67908D8B B8103720 55040083 16E33A30 002846AC BB46A464 BE9786EA F8848B47
09F80 - 453A4FA4 F492102B 674C5858 70801123 30400665 07636AFA AF885876 50AC3846
09FC0 - 13515379 68519744 0031338E 14FC53EA F4878118 76E11027 45411233 9300AF88

0A000 - 5640050B 714466EF 1F855F21 4701A6FD B10B7F14 4005907B 1440011B 9673F036
0A040 - 07BF54A9 F74B05E1 77B57EE0 40487780 8C6E9E8C 236D7DE2 521318F9 62E37DAE
0A080 - 4817CDE5 CC866218 EE23F6AC F8CFF2F7 0B30143F 83160605 D7D415AA 847715F1
0A0C0 - 3514B968 391B178F 2144D58E F1BC832B 08E23EC5 7D84B747 047A1B17 8F214650

0A100 - C14A8CFF 2B754323 400831A0 33C30002 66D01CF1 371348E7 BDCE6E61 3213117F
0A140 - 85B5B284 77AA45CB 8E5AAC83 24002868 4C84B1B4 95F21461 B678F214 485AD015
0A180 - B334802E 01CF8A25 084A137D 78EC7DC1 08DBDA84 98E4D6C8 A6F28597 EED55278
0A1C0 - 90D23113 87BC0C95 118AAC08 F5A21040 075708FF 61B11F67 8F272648 6A808EDD

0A200 - B68E09F7 869D48E1 4BD85A7F 8D5D4102 D2311310 8D51F495 F21438E6 A2E7F231
0A240 - A178F142 8A450144 86D50857 03DB1351 10D5E801 794353E0 68438E1F 1EDBB060
0A280 - 7540D781 776DC4F5 73635D5A FB10A137 068538E0 C1E97886 11AAF77D BC5F5071
0A2C0 - 3576A097 49111AA4 E5448E6E BC594866 44AF4159 F667D876 E27D5111 491831BE

0A300 - B47A474F 0978A076 0E668D67 EA31B38C 370F65A2 DB1BF148 9F8EE38F 5DD9788D
0A340 - B4752D76 6C5FC100 755C49C1 105A814A 31FC9660 01613FB6 56973702 020202AC
0A380 - 3AFA013B 4BF4089F 85A6D002 A4BF3F79 F7C5D590 7F525717 AAF59073 7B40272A
0A3C0 - B4567F22 53686B92 8E528C83 2606DBC8 EE6C65D0 8E736C90 E9066D11 7F17314B

0A400 - 30287BA0 87AD0301 0E0E6A00 30E0E061 5906C3C8 761B7320 B065CEFA 0BF7479F
0A440 - 85B6B6F8 C59B68C0 EB68C2EE 7310460C E82F8F16 17ADF4BE AFB1088E 493DAFA8
0A480 - EC9FA298 E99FAD91 34118D78 E4A3C118 AD08E16F 5F0F08E1 24C16111 0D08102C
0A4C0 - 8E114C31 FF14C8F1 0C108F2A 2207E6F5 096D7B13 7068E838 D071BD55 F2144D52

0A500 - 18E294DD 24901711 5F38E56C 51BC65F2 DB144035 669F71DB 5F07CA0A FAAC3590
0A540 - 79E9492A FB10B721 A5358669 711BAF71 02D23113 8E65FD41 61111311 1A15DF17
0A580 - F33412E1 5D317F17 4AF215D9 179310F1 4D1318E5 19C876E0 86A12137 6A3F739E
0A5C0 - B24D0831 C033F300 68CA7E0F 87B008C6 5EB3F77F 627B6669 6C656038 58ACBA4E

0A600 - 400848A4 E018E272 CAC31F17 5F214310 A8B6727D 1E400112 8B2018E6 B8C1128B
0A640 - AC0760E6 2EFDA131 031F495F 2110C010 1AC3B476 D9013411 01468B61 11438B27
0A680 - 0C914403 94BBF111 8BA00011 BD55F223 7ACF1640 C56F011B E95F2146 D7164146
0A6C0 - DF8BF007 0AFD2550 14431921 32C21325 2E788F57 0D214416 40C5EE01 1001F495

0A700 - F2AC31BD A5F21467 11218414 67702AFF AC7778FD B134AFB8 E39D4D71 368BFF41
0A740 - 3415A016 0B045A01 428A8B02 F758F49D 16429773 F16F16E7 BFE16A7D 91D71647
```

```
0A780 - 491DF793 F4D9AFB8 F234B11B 2B5F2142 8A297AC2 13414696 E50B46D7 164F7F71
0A7C0 - 36134C31 4E90A521 60B8A80F 0719E5C0 8A270D21 441640C5 9EDB94A1 B16E7711

0A800 - 70211847 C017511A FFAC7189 6D5F251B 476F279B E7A6E191 71461341 5E0B8A80
0A840 - D016015A 593842BF 4F4F4165 136134C2 8901DD77 77EDB46C 137D78E7 2077B805
0A880 - 3611014E 96A8516B 1560160A 0E4E0A0E 48016750 394B5114 61118A6A 0D214456
0A8C0 - 079AD167 1461108A 260779D1 6E16F16A 54A8E4BE E7D10500 11014E96 AF01657C

0A900 - 6D1645DE 03137134 DB135011 46132130 C2038D53 4B184057 313A313F F0034B29
0A940 - A08508F3 66818FE4 4A113610 81B58700 7D401A38 AA73401B BA351783 01A8ABD7
0A980 - E20DB10A 13710B8E 67B70011 81348F06 4A111B13 576906C3 014E8D40 5814118E
0A9C0 - 4731137C 2135AF8A F08A9B11 C114B1CD 8E1E4115 178C0584 1CF52F00 8F064A1D

0AA00 - 510932BF B8E2BE64 40D08FF9 91013713 610881CA 3C4F014E 1617D7F5 FE118134
0AA40 - 8408D7B1 81411137 135D78ED D21AF1D8 81DCD421 8F7A2511 491715DE DB1356C6
0AA80 - FE6240B0 0970A004 6CE24408 A2440D68 FE14B120 36F80000 00213414 A1612090
0AAC0 - C5016313 68CA90D0 00000000 00000000 00000000 00000000 00000000 00000000

0AB00 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0AB40 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0AB80 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0ABC0 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 F86AF296

0AC00 - 8F6D0018 6AF0B68F 18114E16 11F098F2 15507781 74201097 A6172024 CE64248E
0AC40 - 7B5E8E83 548C9794 AC386053 72EFAC3B 47750117 FD71371F E95F2143 8A201137
0AC80 - B4774E0D FDA1F088 F2AFB155 41C41451 C4141861 8214AF03 40200096 6D01617C
0ACC0 - 7F75A01F 678F2145 14717414 31741534 17F15748 73B0E487 140E6861 41850DEE

0AD00 - 4E4B44AC 2A4EA4C4 B31CE948 B08E6CA6 5E31411C B15F91C0 15F517B1 570812AA
0AD40 - 286300B2 602D080D FAF289C6 F80F0028 CCE6E8CE AF08EBAF 00441F8E E2014708
0AD80 - AC718637 08A8D0D6 F2F68A20 06BC270E 78EF3A65 7B100010 42084014 A31D7966
0ADC0 - 80850161 84114A31 D2966508 518528E4 47413240 1842D57E A48B2831 FB88F2DC

0AE00 - 141AF21C 915D91CF D9155713 28438E13 B4550853 13201312 366F1692 F8531360
0AE40 - 61361FB8 8F214710 A87223D5 8EDC64D9 13515373 10E9E6FB 8716531A 0982A030
0AE80 - B9E65411 88E5BA44 5A112AB8 7F141191 547860A0 72957E05 0713414A 161311F9
0AEC0 - 62001810 3119861A 0AB0AB22 69125084 3208E237 0D713706 D8787446 3787417E

0AF00 - 17E17471 445B1070 6EB1438B 2D0C98BE 606A0F72 7442D071 35D411AA B51288B2
0AF40 - 518B6D41 CA12915D A6570100 E2DFEE13 3CA130DB 8E6D9456 06DEDDB7 E7578A51
0AF80 - 36DADBAC 2128EA60 30EA137E BDED7C21 08137C21 347925FB 73751361 20DB1087
0AFC0 - 1C477F31 1AAB58E8 55487021 871D0119 15476BCE 873B1136 13574261 33EE1378

0B000 - FFA0B169 AE8E8AFF F98F779D 87005D48 62B13133 8C963E31 CE966FE1 71143131
0B040 - 14332C18 9363E1C2 15731FDA 7F215538 C7E9D8C2 BE071EBD 230F8B3E EDBD5C9C
0B080 - D5BFDAE4 7898DBF2 BF2BF232 C181097B FC739D11 A6AAF31A 098200B6 69EA0031
0B0C0 - 0E9EE000 18E8544D 61324000 68178171 30100182 15631097 DA18B2B0 8606067F

0B100 - 0130D314 A70AF402 1361358E FF40DED7 137E2134 8E9A30D2 3131C311 8135AFB8
0B140 - 12812D58 6040E517 A177A6D4 A114B7C4 F4CE17A1 47EB1455 1E119D58 60631F99
0B180 - 5F275701 371C21CF 7EE1D231 84133CA1 31FB7450 72C16310 FB8EF630 1C1147CE
0B1C0 - 14D171D2 3131D714 7CB14511 81341821 F995F214 3D23131E E7CE2071 34022D14

0B200 - 7CB14517 40C52F01 070AF301 8F1FE95F 21471C41 458407D3 14508503 18F14A96
0B240 - 6B686080 8CDD3C72 40BF0BF0 1BD66F23 19115861 86A6E55F 70207820 134EED77
0B280 - 4521F995 F22C7E6F 6ECD1FDA 5F214301 1F8A5F21 4701791E 710C47F6 7AD8E8B2
0B2C0 - 4AE2AEFA EA4A1156 31629310 0B6716FA 6C5AE343 10007902 73327372 AB915431

0B300 - 62AF2313 110B7E50 48316F75 0018F037 25074E01 7E17E174 77D07310 5607CC07
0B340 - 9504FD13 3031C414 3348F5F0 1798A200 3416EA18 A200038D 670A11F3 A5F21471
```

```
0B380 - 35157417 01471748 AE0FB465 AE0117F1 7F17E14F A6E54003 1C414713 3C253C1F
0B3C0 - 7B5F214F A6E5D111 810BD913 416A1461 32EA1300 3D910B13 57622EE1 33CA1301

0B400 - 1812BD56 71F14313 61348BE0 034495F2 13313514 71331438 B20011BE A1410331
0B440 - A0182A6F 4B218F14 A982EE30 D9E22EBE 4986AD16 A142136E 21341187 E7078A01
0B480 - 36DA1C11 4FAE7A6F 8EDC0411 915CA16A EC34B000 06A1016F 16214A98 23111816
0B4C0 - A142CA14 018A30AA 6F5AD021 F995F214 7132EE13 0DFCA131 DF8D901B 1D71F995

0B500 - F2143EE1 351368F2 71B1DB8F FA0B1136 EB134031 F995F22C 147EB145 1740C52F
0B540 - 17BAE431 F50E6631 14B6A147 EB145176 147A6C51 F03EB145 1C114FAE 7B6614D0
0B580 - 1B8EA06B 8A80D03B 20022101 90602232 00020011 573AB7A2 7A27A272 030FB0F1
0B5C0 - 72D215F3 173D015B 392FB0E4 90B02E69 0B9121A0 F590D0E4 46290FB1 5E1DEE4E

0B600 - 4C42190F 90D2E658 08EFC162 002788F9 0BF0A8B7 95F8E5B1 61731470 3118E317
0B640 - 04478EA2 C1693611 78C6416B CC68B211 7D274152 EA1C69F1 117ED64F 38E3BD06
0B680 - 406117BC 64C28E61 E061F511 78B64918 EF821460 CECE8E88 D061D569 65117496
0B6C0 - 45F8E952 16460117 566D8137 C9135136 D715A672 56D9C870 0ED4BF0B F0A0C1CF

0B700 - 15968CEC B08117C7 644A8ECA E1696581 177F5409 8E26E161 B662E5AF 22E34101
0B740 - 0502AC20 51361A99 40136AD2 A5E02AF2 04A2E05A 5E020072 DF464000 5AF12EB0
0B780 - 5AF01321 B0949913 28228217 3E46D100 0719F431 007FAF4A 0008ED51 18CB5A36
0B7C0 - 06411156 4160AC71 5B6B04A6 49680210 97F65129 80DF1294 FC8ED3C1 6D607945

0B800 - D6137C2D 71377825 137C2061 36DF1372 48B6D022 8A26021D 680FFAC5 8F7C1B1A
0B840 - FD5F0249 F0402180 FFDB1340 71351CF0 E47BCAAF 25702E30 16E65117 DD404491
0B880 - 2EB9A50E B986ADF1 173C4044 A62E0E1E 57E117FA 4044652E A98A1D45 0A910E11
0B8C0 - A1E450A9 20E160E1 866BF811 7044452A F28EF6C1 580AF64F 0AC62E91 85030162

0B900 - 7F6D1381 17F0444F AC015178 C3293001 371F2C7F 27FE3135 AF64EC61 1516221A
0B940 - 0F90F331 637EC340 07554470 8AC00D21 5E31638B 60092F60 E601CC01 7ECF4221
0B980 - 7F10010B 7EBF4001 1B8EE4E5 118CA031 7F767304 D0CC02CA 146E2132 EAD81301
0B9C0 - 11E40116 11461342 01563AB7 31AE9870 02198B0E 13215601 32907000 3203102A

0BA00 - BD10A0E6 596E5030 DA0E8E93 F3130211 5601BAB8 F21540AB 7011BBA8 F214011A
0BA40 - AD22F308 AF520A8B B06AF250 02A35EFF FFCAB918 F1547AF5 8E25A38D D9AA0787
0BA80 - F1BC58F0 7EAE4F07 0AF111E4 6E90A8BA 0E0A31C1 7CC104AF 1111E486 2E7A4DA4
0BAC0 - D86137D8 33FFFFD3 17B8CD79 388128E4 4A310146 9D1AB57F DE4A6A27 A27A2778

0BB00 - 3E1361BB A8F21401 36498AC1 B0745430 DB0F446A 0F521F07 D7E15678 CCF63D6C
0BB40 - AA0F540C AD6CA7D5 E18117F8 CE47367B 11872302 1561D716 7C68098E A6C5703E
0BB80 - 17F8C788 3D6F0C6B 07440D67 31E132E7 17F81731 766C7088 128EF693 101444D1
0BBC0 - AB57A0E4 592E320C F132D8AC 51560A85 13220903 50A45156 78E0C831 11E464C2

0BC00 - 75FD7B2E AAB31F94 7E30C51E 88226800 10BF5D97 B0056068 EC6AE610 11321BBB
0BC40 - 8F214016 414C1A99 5F137144 048EAA86 83201BBB 8F214213 08310003 8E1E908C
0BC80 - 41D076FF 67510520 BB1A0D15 77985011 7F153798 0E30304B 06B0696E D5171157
0BCC0 - 717F10A1 57717F15 37980E11 00AF005A 2C120021 0AAF2A2E 12A04B04 B0496C71

0BD00 - 17115371 7F100153 705028D6 CC710520 BB1A0D15 379806C0 38FE83B1 15B620B0
0BD40 - 4A6496CE CBF4BF41 7F037950 703F4001 088EEC82 5907F0FA FA1208EB B825907C
0BD80 - FEAFAAF6 11003702 0778F400 8E898250 079DEAFA 038D322B 1D71371F BD6F20B1
0BDC0 - 5720B135 DB038117 ABF4878E CED167AE 8C654381 170AF4E5 8EDDD16D 8E8117C8

0BE00 - F4A48E0B D1697E88 227E3F45 38EC5816 46E81173 6F7C7FAF 28319030 F66AF17D
0BE40 - 14932F02 607963DD 811763F4 4F8E6B81 632E8117 22F40E8E DA816F0E 8117E0F4
0BE80 - CC8E7A81 6BFD0034 6E7F2136 D7AF0AF2 14A328E3 8E8395D2 18114ECA 7B00DBAF
0BEC0 - E8C65338 DB13B100 34CE7F21 36AF015A 38EE6B66 6DF8117B 9E4868EB 501688D8

0BF00 - 11778E44 58E53018 E2D416E6 D2031B08 C274D411 7C0E137C 21351CF2 5A8081C7
0BF40 - 48F6FC98 117F3E4C 08E62E06 C2D64CC8 11772E44 F8EB3216 41D81173 1E40E8ED
```

```
0BF80 - BD0600D3 104639F8 0194AB38 E70FD8F8 75C18E57 0E4CD137 8E83FA8E A0041711
0BFC0 - 33E2DA8E 87CC6A6F 136D71B4 95F21461 64AD0142 EAD2314D EA540D08 1C6DBE00

0C000 - 3F000953 56295141 30653E84 42380DF1 3610BD18 92027CEC 17F737DD 84C0D182
0C040 - 194D60CD C57AECD6 133CA131 CCCC1017 5DC137C2 10AE9137 E0466EA4 16AD1D88
0C080 - 1DCE4458 0D0F6D75 D2D90613 70613716 1171DBD5 8FFE1B10 713507D5 562CD119
0C0C0 - 13445114 A1C114F9 620CCD52 F11A1351 1AAF0DA1 371CFEA8 1C71DDAF 61136A3A

0C100 - 88227C4C 4037B666 47B88227 93C4D170 76616B88 22762C4A 07A966E4 B8C5EAF8
0C140 - 8227D0C4 1F7BA765 3B811743 C4FD7F55 72307443 6B1B8117 A1C45C76 B3690B81
0C180 - 1780C43B 73357600 8CA15F27 8E5091AF 7AF2CECE 01D3AD01 371F495F 2143135C
0C1C0 - EEA81C45 18C282DE 447F1C11 4DE714E1 61965AED BC6BF2BF 230F635C 8812D3CF

0C200 - 1564160A 4E7E0B78 9B4B0D09 4D40CC17 FD6DFDAA 4E5FD8AB 40CFC4C7 136DFD58
0C240 - B040D41B 5A8F2140 18414418 41761471 441C6157 7B065273 0C816A06 450B4690
0C280 - EE5AC5BF 68B240DA 8B540D5E 017F133E DC2C0137 D58F601B 1D9BF2BF 230F1CF1
0C2C0 - 5571BB98 F215EEAB 5AD5DB13 48CC36F6 53A0F142 8E54EB4E 2C9A35A3 5A35C9AB

0C300 - 81361621 42136C20 61631564 16003203 1226BFBA F2BCE550 AF22EAF3 B075D282
0C340 - 1822AF2A F32EB07A FDAFFAFD AFE66517 1D584B82 18222064 62822AFD AC2AC324
0C380 - B04B04B0 4B04B04B 05B05B05 B05B0597 BE4AFCAF F97B348B 0328B4DE 9FF32AFC
0C3C0 - AFF5A189 09A89187 67E09407 0BF2CCE8 F9CD4A0B F797F5F9 40528325 0B77B7B9

0C400 - 7E81AC08 6A0186BB 0BCC550A 7BA0CA0C A0CA0CA0 C2EAF56D 40798276 05822821
0C440 - 216091AF DAC2AC3C 0B4C550B C8AF12F0 CBF56600 A71A0F59 F88EBEAF 9AF79497
0C480 - 0E4BF52E 9190190D D0CCB916 5FFD0208 0FFACD03 78848218 22226E11 AFDAC2AC
0C4C0 - 397F6397 EC020317 06D81B40 550BC820 3180D0AE A3200FDE AF1BDD23 00E0B405

0C500 - 50BC89FF 70BF2CC2 EAF16600 B05B7B59 FA7BBF20 D51F6990 7B818218 2204B24A
0C540 - 2C055E09 6C0094C4 101AC1AF 99797694 8C02031A 065F0A75 A75A71D6 C6AD2550
0C580 - BDED6AFA A76A76A7 A2090C50 BF5BF4AF 22E305AF DB95A0DB 05B7959F A79BF20D
0C5C0 - 58E82297 A80A3EBB 603AC1AC 304B26A2 E4F2B24A 2C054606 6DD96C00 89233890

0C600 - 0097F922 03101615 005922F2 96EE1892 D1890419 79DDDAB4 A500BC80 16D50D0A
0C640 - F14BE96E A496C008 90338917 D2031E0A F1AE5AF0 203210FA BAD90C81 58858FBA
0C680 - D2400203 1F09461D 0196C01A FDAFFAFD AFE01AFF 9A1C09A5 009D120A FF4CD01A
0C6C0 - DCAB8A24 D0540CCA B4AB104B 24A2C054 C0AC580F F5B4AD0A A8027540 89F00919

0C700 - 00A9188E 40D00371 F55E096C 00D0AF10 178100C0 C4B0BF1C C65FF605 D728F72C
0C740 - 544320D2 3131E244 12190A70 2980980D 001E6401 34000052 E8BE002F 0176B185
0C780 - 87231AC8 76C04005 F57E9184 872A0761 1AC57AA0 42187914 979C3B7D 563AF711
0C7C0 - 88790095 9006A9B7 1618587D D0AC8717 0473AF4A 749FE901 10639C97 2C0110B7

0C800 - D6110110 AE381F96 B1EBCC5B DAA3128E A1288A8D BE447011 101BF211 110047AA
0C840 - C1859942 00849BCD 01101AC9 AC1AC310 897934AF DE09FF70 CCBF2D8C 5AC4AF5A
0C880 - FF472101 550BF1AA 3B27B715 9FA71CC5 BEA2F03A FB111949 50BF5028 218228E4
0C8C0 - 9C056007 0175345A 0078C47E 07234531 10807118 8780063D E95F005A D77307BA

0C900 - B8314001 78ED89F0 082201AF 2203220F 018E984F 7800763A 6360D707 738D06DB
0C940 - ADFAB7A2 6D2540CE ABBAB304 B26A2E05 5A0AD2AA 702AC780 FF2E91B0 190FD0CE
0C980 - B9365FFD 22080FFA CF038314 27140893 12894C1A 83719124 90B61A4D 401227E0
0C9C0 - 07A61700 26D42228 ECD3F7CF C5212084 7AFDAB6A C60180CF AC57AB01 038E04B0

0CA00 - DA203410 59495D80 AF35958B 281FA8B6 358E50B0 D620ADB0 18E90B01 13DEEE2E
0CA40 - CE4A0B95 0D55FAC9 80DF7950 211331E1 000133D8 97F1CD26 CBF85733 2000D522
0CA80 - 87A3187B 02AF2AC6 8CECCE87 BA094EA0 59E94E4E AD28CA9C EAF9AC28 4791E708
0CAC0 - 32C28578 7AC487B8 0A16490A 92631091 E7083222 A92B9E2E AF7AF6B1 7570E6B0

0CB00 - 720A8301 AF781F90 BDD54D87 BA0948AC 56B9481B 5FB639BA C1A4D880 7086700A
0CB40 - FF1331F9 F6F21471 3380CFA0 E4E0A0E4 04A0E461 880A0D23 151D5AC1 6800880B
```

```
0CB80 - 080DFAFF 01877D08 81FEAC25 9E2069BF 890ED894 DC857AFA AF289160 7FEEACBA
0CBC0 - F7AF664D F10BAB28 77A0890B 3580B268 1E80C289 4A088380 A2EA2613 31F8D6F2

0CC00 - AB715730 E3F15531 3111B019 4DC0D920 8C877C89 00004201 0B137D7D AD231478
0CC40 - F7C21049 DC213511 9AFF1108 F3B51013 52E07145 1740C55F 1B109F2D 231048EB
0CC80 - B6C133D2 3147EA75 60109140 AF9288E2 27C70501 11140D5D 23147C91 341F149F
0CCC0 - 2D231048 E528E2E1 84146060 C55F1361 358E1775 100AFB10 9056E1F1 B995F214

0CD00 - 60104B24 A2C05010 4B26A2E0 501AF9AF 7D8BF797 B21E555F 2EB078CE C7FAF501
0CD40 - 77ED8217 5765504C 384AAC1D 6E6C64C0 8EAC5F65 30948808 5A56072A F6A707AA
0CD80 - D8217836 53194800 2031D08C 7C8F84AA F2AC197D 412031C0 AC0BCC8C C27F94C1
0CDC0 - DD68AE15 2EAF9AF7 A0E97EC0 822AF16E 1190EB0C CBF265FF 714F2EAF 2108D6AF

0CE00 - A6D00898 B70DB46E 653F471C A570FE85 A2EB99A0 EB06AF4A C76600BF 5A4F59FA
0CE40 - 78A4D53E B46AF8BF 10D8866D AFAAF220 34299998 ACC2100A FE95EB0C C7080550
0CE80 - BF57E80A F6AF7110 27601026 B0427AF7 BF57C327 B5095863 88FDEAF2 20A89A71
0CEC0 - BF57EC28 7A50B7D7 3308A8E0 BF553FBF 549094D7 FCF7257A FBAFA86A 50BCC8CD

0CF00 - 75FA71A0 C59FA80E 70C01AF9 AF7D8A77 A77A73BF 797B31E5 55FBFBAC 3E469FDA
0CF40 - F50173EB 82185B77 846D0071 DB82184B 7C545419 4880AF0A F16E9282 1AF21088
0CF80 - 5A94C508 4AAC0AF3 D6AC2C65 82D6949B 0E4BF569 EF2F0D89 6B0E656F 6E706180
0CFC0 - 120AFCAF 22C33486 5AF77AB1 CEAFDD22 5CEE68E2 FA057F8E 25B0120C C1204B09

0D000 - 0A3E62F1 B96AF811 0AF3D777 C15C101B 07B7159F A7189701 BF1CC0D7 8A046E86
0D040 - B21AFB10 878CE118 AF7D62F3 07268AB5 0B0795BF 40C90B42 A0FAF4AC 7550BF4A
0D080 - 4E59FA78 B45ACB6C DFE6BF5A 4E59CDBB 05DAAC08 6A808E29 2F6E51AC 0DA86A90
0D0C0 - BCC745C8 6B607903 979A0740 067658C1 52FAF289 ED3A5E30 4B5689D6 489C6589

0D100 - B4689A07 899A7897 D0203233 32802203 03270120 3E549955 08174139 62E01203
0D140 - C9423408 9710132D 01203A80 86135803 32C01203 83353803 332B0120 36380333
0D180 - 32A01203 43333329 01AC2203 E5049929 05852032 017D798E 125F94CF 08EF0905
0D1C0 - 60670477 BB831400 17DBFAF3 AFF8C5D2 F2490BD0 A0F90F90 B072E03A F22EA1EA

0D200 - F5243018 6A40FAAF A86B007F B18E701F 73C10270 A38EA02F 84A94850 85AD2203
0D240 - 048B62B8 11811CC5 AFD9D777 8F400DBD A86A40F8 AC0AF1B0 55F98EAB 6F821822
0D280 - 73D24008 4B848AC1 AC38EC34 F5508588 E1B6F7E7 19486068 8097D606 2E08ED08
0D2C0 - 05C27451 734A4128 E2C3FD78 E714FDBD 38E0B3F5 606B3F7F 8A76218E E31F8313

0D300 - 220D2312 18789031 115E3AF0 AF12EB05 0094F606 83CCCAFB AFD624C7 3D0D6C65
0D340 - 5178C097 91520319 0644A8E2 E3FAC388 FA08AEB1 57091D4D 0CA89A06 A06A09A8
0D380 - 77980AC0 90BB0712 FBCC016B 1F7E7020 97FC0D23 160696F9 4AA03150 6DF9645E
0D3C0 - 822077A3 94600603 96C00060 2100AFC1 01AFC011 20AFC121 AFC01119 AF711801

0D400 - 102AFC10 3AFC0112 2AFC123A FC0111BA F711A011 0AAFF10B AFF0180F 20680F27
0D440 - 33007400 88892AA7 2580F20E 2792A71D 2203141D 5248478E D45F0372 92AC780F
0D480 - FACF75A0 74B0287E 625A4AFD 96C81924 C096DE09 40169482 42180FF0 E43AFD7B
0D4C0 - 607A70AC D80DFAC2 ACD94F20 AC3017E2 8AFD5219 6D9C9492 C246FBF2 E91AA091

0D500 - F0158A91 F3E2265A F944898B 0398B4EC 99798993 4C50E24B 04B04B04 B04B0401
0D540 - 24B06B06 B06B06B0 60175B17 5A150292 641968F1 96A008E2 31F0296C 00038ED8
0D580 - 7F50096A 2F7D7402 7A35AC67 56142195 800B0690 80055130 496800B0 6A6C9680
0D5C0 - 0B06017C 3173507E 215C096C 12AC003A C197D61A C0BCC203 1308EBEE E03AF1D8

0D600 - C4AF0570 BCCF9815 E48AD8FB F5CC8220 38218220 176308E1 0FE86960 70108670
0D640 - 0AC2B46B C6010084 58320085 50186500 50E84983 15085984 78320085 7017D807
0D680 - E9F70DE4 00100109 7C607373 57192691 68A097F0 197DB001 7C405FE8 EFEDE7D8
0D6C0 - F79F4729 F11894A0 01192EA9 2AF78792 1E7E72D3 181AFF5B 07584A77 CF8C67CE

0D700 - 8C106F8C 7BFE8C62 2F66CC7C EF8586A0 071EF848 8566D007 3DF84884 674EE7FB
0D740 - F53196C0 02031B08 CE0FEAF2 10884A84 B87850B4 6109AC18 47979508 57948D08
```

```
0D780 - 5B868508 5AAC025A 80D6C655 0A0CAF6A F4879D1A 1E90ED7A 1E5F0B16 12872931
0D7C0 - 2890E567 13379637 56371637 4831287B F25BF7D5 3128A1E5 BEA92128 70037833

0D800 - 869A0765 374437CC 24517393 717340F7 97369EF1 28869719 1EF01287 B0312855
0D840 - 0B1690E2 41287CA2 7982401A FCAFF7C7 278232E1 28AF5128 97C7097A 4190CF07
0D880 - 3C2CD64F FAF5AFC8 79B29793 28AC12AF 22D31548 66B0868E 03103975 50847CC8
0D8C0 - 79312779 D1AF7D28 E17BED6E 65A0BF12 E492D220 308C2411 868808E3 4AE6BA0A

0D900 - E6258098 0D0AF071 91550B44 B7159FA7 10DBF4BF 18872EBF 5BF5AF62 F3177AFA
0D940 - AF32EB07 AF95A1B9 6B96A4E5 6FAC2AFF B7FA79AF 5AC6A0C5 4EBD0958 21A4CA6C
0D980 - AC2BF66F DFACOAC2 A7FF8D18 6870F8AF F8E41BE8 76518311 4BCC2031 40235C31
0D9C0 - 1994AB08 31607CC1 7E3D7B91 8EC5AE8E C49E8EA4 BE8EE49E 7214876C 086B00BC

0DA00 - C0186A00 55F8CA8C E0261194 25686699 04256668 66669999 07686666 66999999
0DA40 - 07666666 99999999 07666699 99999999 07669999 99999999 07999999 99999999
0DA80 - 03349915 29235471 08447933 61893587 026D280F 1D7341FA D004EB05 136DF80D
0DAC0 - 11567DF1 34DB03AF 22EB0697 10001B70 461B7B44 003A7C50 0B74A7BA 7002AF58

0DB00 - 6972799F AF7203F9 91854806 6516903A FF25AFD0 1AF2AF32 D31544CE A75A7750
0DB40 - 0B7501AC A810BF20 1BF6814A C6AC001B F7815ACD AC7ACDAC 1016188A C3732FA7
0DB80 - 6BF6E6AF FD201878 70858038 480287A7 085A0384 A0287B70 85B0384B 02784B84
0DBC0 - 88466810 7A3B8486 A007F2B8 58856821 84784A84 B731B572 96C00866 606E4B75

0DC00 - 7F750EAC A8695085 769C1AC1 879A2979 6478AE57 08A8A386 62330597 5A2CE8A2
0DC40 - 62502868 6194C618 AC11767E 5A04A097 95085794 83185A86 8B084885 B84A97D6
0DC80 - 06F21D6C 65A58661 57D7A8E5 67F8EF86 E8E047F8 EA76E72C E8EF87EA C08E398E
0DCC0 - 8E657F8E 386EACOD 6C64C08E 266E7DAE 6A308AC9 071ED4B0 8663E640 F876117F

0DD00 - 9DAF5AF0 CC4C07C7 EAF0AF16 9908A80E 7AADAF7A 82D6E64F 0B460D88 73F5BDAF
0DD40 - F4A1B07A CBBF4BF4 A4E56FAC 2A72AF4B 7151EAF8 B47BF10D 8879EAFF AFDAF48E
0DD80 - B87E2879 1D6600A7 1A0C59FA 80BF5958 A00C61EF 0CCC88E9 FAC08ECC 6E868F0B
0DDC0 - CC71BD8E D95E86BC 072AD8EE 85E879A1 277DBCAF 7D2CECEA FD8E207E 7A0086A5

0DE00 - 0BCC038E 538F8770 0822A35A B1500BBD 2EB15500 E4B05031 467FFCB5 F74F3856
0DE40 - 6830FB47 F9616F7F F6AF0CE4 C0150716 F53F62E1 8067F69B 5F7BB384 674D675F
0DE80 - 178C37F8 372D17EB 1A82A4E4 9510C16F A0E59F15 3771927F 937DF24B 111B1347
0DEC0 - F4874727 8ACBCC7B C27A9276 F2709104 11CB065D A7B336A3 27263719 27A53866

0DF00 - 01760642 E7EFABCE 720644D7 8927E230 47A2110C 7331AC19 7D92AF01 6F11CD2A
0DF40 - 4E4E1D51 6F150716 FCD56FE6 56E114A4 C4B71041 F678F214 313116F1 6F133131
0DF80 - 068A2527 5C107137 067AC17E 32154707 13777C05 FC73A211 B134735A 74917245
0DFC0 - 7C9178A1 7C021547 7A900713 7618F207 5D073A17 30215471 1CA82A4E 41410C16

0E000 - FA0E59F7 0318E8C3 F1537762 1765B7D4 086650BC C7DB1154 711CB065 DB203260
0E040 - 88EAF930 48CDF9A1 B178F215 67134011 F678F214 71350117 417F038E 8CEB1F17
0E080 - 8F21517D 0810D8C5 C5C8F0C8 326088EA D8346061 801371F6 78F215D4 30AD57E9
0E0C0 - F1B088F2 15641A99 5F146164 14290672 A4E4B38B A2213018 114A988E 018D1527

0E100 - 984E08CF F70130AF 01517755 F13253C8 BAE03153 208C762B 7B2F771F 10C15277
0E140 - 8E450074 A0AFA51F 153717FA F8143542 157717FA F7147541 2EA9C151 7A9C17F1
0E180 - 411CF018 56AF2AF3 2EB07866 50BCE8EB C4F8E0C1 E8ED84F6 B00058E5 82E83140
0E1C0 - 0203247E 508C8C7E 83100581 77DF8E7F 1F755F79 897FAF13 71F698F2 1451E995

0E200 - F1471357 9BF1F698 F2143131 042E8228 218CD8BD AF98C7F6 C8E4CFA1 3514A042
0E240 - 1B044008 CD3F06ED 613610B6 9271328E F4B51324 008CE631 76D274AF A4E5D079
0E280 - DFD0E45C 17076759 64028F32 2B14708A CC111AD2 8128B6F0 03968807 073028E6
0E2C0 - 84F02801 84658080 1856789F 4F070238 66607E02 6AE08017 D7F422D6 70F27612

0E300 - 75F27402 7B7E7AE1 8E313F6C B0811792 2D2F5A89 109F5A89 10872817 1F570B57
0E340 - 1867C6E7 6367E4E6 38062B58 8227BE17 4954FE10 01097CAE 8EDD1F4C 51107D0F
```

```
0E380 - 42512173 0F484100 1197A627 1951187A 42766172 851197B3 277517B9 576FD8E6
0E3C0 - 62F7AB59 5D508227 F218E954 877ED8C9 5E06C357 9DD10A68 601A07FF C55F713E

0E400 - 13606784 17C7E45D 10117F7F 6E48C100 7DA18E72 11163118 F2DA119A 8A148787
0E440 - 045A7C7D 70DD10A7 B657D6D1 08071361 10151711 21CF1517 8EECAB14 A20311F9
0E480 - 66331618 E8A5176B D8E91312 08F15681 8FBD3B18 F9AB8117 F6EBF619 B7F5D702
0E4C0 - 17764119 70217A20 78541117 42172A47 0DC75441 10711172 6474AC7F F07E007F

0E500 - 748EAAFD 62BC95D7 08A8127C 0453296C 82948322 031438EC 21E02203 15341F95
0E540 - 97094C1E 038F785C 1047A709 3EC02031 3365CB8E 5EF04FE1 46A2656E 80D31651
0E580 - 46F6E6AF AD280CF8 090D5AFA CE8B2FB1 02182713 0F620985 CAF69855 A1671461
0E5C0 - 32EE1341 12DE1020 3201BDA7 F2146D50 11111188 AE71DA8A E01AF054 37F23D0D

0E600 - 2048BA40 DE80D0D2 0D480809 57FCA11A F0C21341 527058EC 80E50020 3220F9BE
0E640 - 40033131 64EE8AD6 FD145F77 DB5511F8 F1E21438 EA58D431 8EB17D42 4AC09789
0E680 - 2D1ABCE5 97851D9A 365D093A 80BF458E AB4BB0E4 1FEF6F22 0159E619 96C42000
0E6C0 - 5136061B EF6F215A EAF1AFC3 F7648290 31158200 0550A70A 0E59FBF1 0C54F158

0E700 - EAFC2E91 911CC90D A0CCB915 5FAD4071 34AF68CF EA08EF8C E5B09480 0638B8EA
0E740 - FFD91970 88FEE959 7094C4EA C0AC1D2E 6E68BA31 AF0B54AF 1E5F56AA 0BF5CC48
0E780 - 0BF157F8 22AF3AF2 2EB06AFD 94921E62 0A05BF55 50B75AF0 2DB035B0 A70B0759
0E7C0 - F2EB035C 0BF37550 4C094A11 BF4E6B43 5607E30A FC2DA0E5 8A2EA0E5 0AA4E5A9

0E800 - AF1A9CDA 20340030 08A2C032 4328A600 A5D02E62 0A04BF45 50B74A70 B4759F01
0E840 - 81174A07 0EE4C481 176908E3 BED6C308 11763041 3948C28E 3BAD6320 8117D104
0E880 - 8194C318 E3AAD6A0 08117400 604B7B40 8C25ED88 22794026 6A102031 417F6C49
0E8C0 - D88227E2 0234A382 28218EC9 BE4DB898 4D8EFADD 5FA7D297 A00673D8 CF54D8E9

0E900 - 14D5008C D04D7C09 79EF7E00 50096800 8C7F5D8C BD3E7A90 B060B846 0B154013
0E940 - 4AF9AC61 54773901 40037270 A87A0E55 03031540 A8B13415 67AF7796 0146AC30
0E980 - 3200480F 1AE77A30 B0B0B846 8470B134 1567AFDA F7743014 6AFE05AC 4AC10321
0E9C0 - 62CF226C BF1B669F 22034109 F2211560 0113E0B8 560B13C0 10420D11 14D01041

0EA00 - B8F2E280 8F14A14E 902AF14E 21A0EA0E B3514A96 67FD0B35 14A966AE 80801148
0EA40 - ACFA20C5 D4F0C5C0 D23131EA 1B779F21 40038ED6 F2404F14 A310E962 A0B66966
0EA80 - C131EF4C 08EB6B46 6338E86B 455F76E6 72D34811 0178C348 01195221 114F0AF2
0EAC0 - 2E32502A FAAF22D3 55209997 40063BF0 4B26A2E0 5AF74A12 23049A37 0923B0B3

0EB00 - 7B37B378 E5DF3AF6 78C1AF0C C9F650AF EAFFAFA8 E5BF3AF6 78A1AF0C C9F640D6
0EB40 - 8AB0074F 1AFFD520 31EF8E2F A44001B7 79F2AF2D 9D7AF014 2BF07301 20AF2316
0EB80 - 6B7A550A F0BF4B74 81CAF2A3 E9FE40D6 D729A9B7 181AF014 2BF07301 AF2DBF2A
0EBC0 - F581EC18 1EF2BF2B 71BF1203 5837C00B 797590AF 8DB7E41A F720317E 8EE4A4AF

0EC00 - B7721AFD AF732F00 2854024A C186C808 4CA4D80F 287CF180 1AB4A3C5 CF80F2CF
0EC40 - 58ED3A7F 50E94950 85C8DC65 20D22032 875D732B 4065DEAF 1D8AF4D5 AF92094E
0EC80 - F09FAA0B F20C50FA F8AF0A0C B04B7159 FA710D40 3BF65EEA F2D6AF0E 405AF182
0ECC0 - 281E8325 0A78A749 7ECEAF4A F9030420 AF1AF0AC A810BF2A 780C4FDA 75AF4A75

0ED00 - A75A786C DFBF4BF4 BF4BF4BF 401BF0BF 0BF0BF0B F001BF6B F6BF6BF6 BF601BF2
0ED40 - BF2BF2BF 2BF20154 76FB1D3F 77827611 8ED94497 8D53048E 3AB38E30 83874941
0ED80 - F244F215 7090E938 FDC6008F 1270066D F07B6F3C E4F15A18 EF43B5A0 70106900
0EDC0 - 8E23B38C 08C91B9F 6F234111 1114418F AF215471 8F15C78C 1874F966 FDCC3F71

0EE00 - E177505A 02D33220 11087FC1 48020620 B1207FB1 52F20D23 1618B65E 1188B2DD
0EE40 - 96A8DEAC E1F174F2 14D1E374 F6380751 01007E00 118500AF AAF2011B E95F2142
0EE80 - 1371358A 2007D7A1 7F94850A F0037F56 F52C3F79 317FAF5D 02C36521 09991087
0EEC0 - 1201207A 101181F8 49F214D1 E649F149 65EEAF22 D3323010 58EB35D8 ED4C3AFA

0EF00 - 73E0500D 1AE88A00 0D0CC030 4201BDF1 E2706050 0041BFF3 E2156226 0C5DF156
0EF40 - 2A26313C 5C08E9A6 469008E6 A641B8F1 E2370087 000815C7 8C02461B 344F2AC0
```

```
0EF80 - 15040315 60A06400 1560A060 1AF46F0D A3F7C307 D10613F4 E46FABA3 F7620770
0EFC0 - 01D8545E 7AAE1FF4 9F2792F5 7096C00E 40262A18 C1CDC613 6F0CF4F3 0C712067

0F000 - CD1946F7 6A3F73DF 71308F02 B816EDFD 71331FCD 6F214F0B 8508510B 0E6715D0
0F040 - 133037D2 E73BE1BD D6F2D220 30B8B650 A8615C00 2A246F00 A3F7C6F7 BFD718ED
0F080 - 6F48A850 30F1FEF3 E215D06F 2D1CF133 EA413109 1F495F21 478B2F11 31121AF2
0F0C0 - 30F14DD6 17115DD1 7D036505 77601479 3A000B84 B0B15531 727A3055 E7640147

0F100 - 93A000B8 7BA10BD7 137DF8E6 292DB135 6CDF0B17 273005FC D2157317 3133CA13
0F140 - 303041F1 75F2147E 61350107 13415631 6293AC08 E29726DE F1360601 8ED70A61
0F180 - 1070341F 995F2147 13568A0A 8980F013 6FA809FA 1361567A C2A9216E 2190D112
0F1C0 - 0313105B 696B4015 63162604 06004181 1371FA85 F2143174 14117414 11741451

0F200 - 37153703 AF2A8980 F080FE13 21B495F2 1CF137D7 146DF8BB 9A137130 15570420
0F240 - D014A314 B9E649D8 161342A5 E1C2A34A 34A34CA1 32146132 C206037D 627DA244
0F280 - 530AA87A C11567B0 F47880D1 88035A0B 4C3E5E5B 45291871 567A9220 90FD0B45
0F2C0 - 70C1645F 16F15636 94FAC1AF 26F3F16A 146132EA 8C9B8C89 F9289E06 B45B4575

0F300 - B1A45A45 80D1AFD8 0FEAFD62 0F20A871 61146D51 3430C903 E130AA8F B0FA0B13
0F340 - 246A1321 816C5F15 6766CE16 1146D513 4156780D 1AC1618F 6D8FDC13 01CF3449
0F380 - 5F213714 38BE3613 517FD2D3 66B07741 77814FC1 56780D18 9014AC1B 45B45B45
0F3C0 - 891DA89E A1166D21 5E3D7184 1461346D 20161146 1346BBF6 9ED162D2 15E3D716

0F400 - 7146132E A132D8D2 15E3DF06 C734495F 2136CB14 21341631 CF137135 17FE28BB
0F440 - FADB8E8A 0C183071 C315D32A A9235EFF FFDD9136 AF51C414 5DBBF281 21C615D6
0F480 - 6BBDAC72 F33850F2 00E47B43 AE3813AB FB03580B 8FABB94A 00B46018 E764B16A
0F4C0 - 132EE132 8DB14B1D 114E161A E531020E 6596A00B E1BB114E 161AE532 06D92542

0F500 - AA1A2D9E 551BE1BB 114E161A E531060E 69B25027 4AF1321B 7B5F214E A6E451AE
0F540 - 71611461 347F0050 074205D0 0216FA6F 40015631 62935DE1 32D81320 3D231F50
0F580 - E65D7A66 A33A36A3 3347F3F2 CB13415E 6AE726B9 6B961342 0A6F010B 1FEB6F21
0F5C0 - 55303132 1BEB6F21 5630B132 038C17E9 86F607C4 47B8B7FB 1161708B 20136061

0F600 - B178F27A F14FC309 98603A4E 50615071 031B2C7F 21507048 6F606564 75F20713
0F640 - 403B04A6 496C15B4 6B46B465 1475505E C6DE2730 3153764B F762363F FAC5A4E4
0F680 - 5EA4E4BE 31968E4A 5C560679 F61B2318 66AEF070 7D231520 203BF4BF 41B178F2
0F6C0 - 1468AABE 13706D21 5F3C6D5C 0164146A F3D7CA16 41468B14 0D9EA25A 802081C0

0F700 - 616A1468 B6B91593 C4071738 BC65137C A131C913 4DBDDE35 60D3D981 F8E3BDB3
0F740 - 1025801C 114DCF57 F0713416 1142DECA 13016F8E 68DB6430 133EEC91 37C8DBE1
0F780 - 07134161 14616D8E 11F5D913 4DB8E40F 51F995F2 14713503 17BD215F 31F688F2
0F7C0 - 145AF91F 178F2155 71EBA8F1 5771E188 F14517E1 55401000 00000000 00000000

0F800 - 6A4134E0 010D7156 78B37013 4038AA8F DE963BDD E8AAAED5 A46432D2 78014008
0F840 - E67AB312 34901366 4D06C678 413AC00A 000A0001 B098F221 15601F58 AA0A465C
0F880 - 285117BD 23021891 42144184 140CE331 00294A50 30074A01 09550175 A0E90A50
0F8C0 - 172AF215 73108207 1704008E FD9B3123 4E68E55B B8EFCAB1 F188F214 3D6F02F3

0F900 - 00861B0C AC4C430D 20D91341 6A146E21 32EE1F17 8F214513 41F995F2 14313115
0F940 - 37038DEA 21031F16 0761BCD6 F21562A2 6AA2500B 2602058E B4DC8218 22298EB3
0F980 - 0D10BDA1 8915E915 4716F150 301058E5 F3C8E09D CAD48FF1 0B110304 AF22F318
0F9C0 - 5200E469 B2453200 59BEB205 84A8E290 D8EADFC1 0B04AFA2 0306968A 23045423

0FA00 - 200F9B21 1307A0C5 11306A5E A8AADAA8 68160E4E 18915A91 50703136 1BB88F21
0FA40 - 44134015 B85F9E83 FAFA35FE 10B515A5 9722184F 35FE1092 9765085F 7DB4540C
0FA80 - 60E0A87F 40D21550 8C6BF867 6386D9F7 69498A0F 779359E8 FE2C104F D1B178F2
0FAC0 - 1468AA0D 8F674A18 E750ADBE 21371CF4 6BAF2154 77C138A8 6A848143 8FCD1505

0FB00 - D11368EC 20A16114 A136319B 966D1336 4E48FDFC 208F9735 08536D00 8F229508
0FB40 - 438E4EF9 15671098 63901631 563D5161 14A8428F 670B1411 852B2516 114A550A
```

```
0FB80 - A1840314 2966C085 03000E69 873037A8 9844310E 14A9E2A1 31FF9665 08541611
0FBC0 - 421305ED 13610872 F2310214 C1617E82 7F231191 5C916186 35016386 25016186

0FC00 - 05016187 3D287482 11813514 7D5A25A2 5A2531A0 985B0BE5 A6D56061 D0318214
0FC40 - C1611331 F188F2AF 214796D2 6DA92940 E4340000 18B29411 013117A1 43137E21
0FC80 - F178F214 310017F1 7492D20D 1440CDE5 143E4E4C 4C21108B EDED4644 0131172A
0FCC0 - F015B3DE 92DD0E68 E6AFE5C0 8ED9FEE4 E520D910 8736231C 214C1611 10725231

0FD00 - 9214C161 8709131D 314C1617 351860F5 69D01F67 8F2AF014 3174AF21 478ACB0D
0FD40 - 5E5E541D 10831B58 1CE472F1 110D6E6E 64D031C2 81C7BD13 1D514C16 16D9F1B1
0FD80 - 78F21527 1308EB68 481030CB 02DA30FB 028ECD7B 8458438E 49B48753 4863C031
0FDC0 - E01C114D 8F941818 EF61AAF0 8E35FB81 C8EA3998 EF0848FB D3B18EA4 1A740069

0FE00 - 388D2A22 01FB88F2 14313101 84A86F00 86D0087A 007B010E 06A0C400 77004400
0FE40 - 2038F388 6017E143 F48D6C3A 17760370 247F6021 5C716731 FF14C8C1 98913610
0FE80 - 913414A9 08F11618 F7887052 18E89C98 F7887049 116115A3 D87D50D4 7C907EAF
0FEC0 - 614FD220 31A37540 3F452716 365602C6 96154716 F35E6560 215C58EC 3C98F078

0FF00 - 707D0016 A16A8D51 150D2DE1 BE95F214 6D7E2189 1421308B 2008C5A6 F1F0B7F2
0FF40 - 2030214B 0114C161 8E95DE04 8D221508 488FA777 08F92D70 8FC27A18 F7605073
0FF80 - F1D58689 08F53410 7250D150 1133D813 37C80EDD 910B1338 F388608F 7F310500
0FFC0 - 8C1D39AF B7D34F0F 0AFBBF6A E2C203D2 15E3D521 7660D013 04431711 37135840

10000 - 8418BF62 15B38A00 08B0B113 31301317 E005FD17 18518500 3173D014 B137C213
10040 - 514B171E 6E690C6E 018D9997 08538426 90084385 279E0521 8E494292 D28C9351
10080 - 7ECF4758 73901716 870D2CE4 D1852853 69008428 4378A041 C7CA07B2 F5D1872E
100C0 - 3137D513 7746F8B3 83D45038 61908D21 470872A0 860D15F0 8F0F6701 7F170132

10100 - 8A850131 8F96F408 EA4171F1 C6F2AD01 43EA81C1 00D9068F 6FB1007D 58E7D591
10140 - 188E9759 8F898108 DE730078 AC8DD177 01B8E7F2 D215E38A E0001135 D215F31B
10180 - 8E7F215C 3038DE92 20D52034 004F28B5 D673E28B 50031EF8 E7F70500 171D230A
101C0 - EA4A415F 6AE2E140 214F8E74 BE7F22D6 F2F28E62 BE8B9201 5F5AE25B 0C11796C

10200 - BFC125A0 E2001038 40843047 7C7D2A6E 10C808FA F1E51FB7 4F280A13 32034E45
10240 - F28B6B21 33D2AE98 1E80177D 080696E5 1B55AA1A C1A6558C 6D22DA10 B24A8520
10280 - 80F180F4 8E2BAE23 A99F6F6A C9812928 05B24593 15D9179D 22430476 70805A05
102C0 - 295B0B25 AC1A4DB4 57B602B7 5606A5F2 432FFF72 702765DF AF7D2243 08723080

10300 - 513415A7 2037B3DD DDDE2791 2017E207 930256C9 F702051B 75005BF0 18D47700
10340 - B055000C B0503D22 4308804A FB0328A8 2AF7113D 22430477 CF5BF805 9891278B
10380 - F5BF8062 39160124 A8528B07 6ECFAFB1 5D917924 A050122A 6E400D5F 1F1AE513
103C0 - 71351341 5B9169A6 D45115E9 99A0F158 9AFA56E1 599F5F5D 91796ABF 11813576

10400 - 90AF701B F6BF6BF6 AB080D00 DAB2E6A3 60C5AF25 A3AA0E59 F2002D2A EBBF7BF7
10440 - 133C6CAC 6C6CA133 DB0170AF 7930D760 5F0C4009 0D8F80C2 0C4D0909 8F80C503
10480 - 2332CFF0 21F2B5F2 14701BF6 8C298EAF 92034B74 F2108277 1AF714F2 032003AB
104C0 - A172A6F4 B115F517 01592178 773F47E6 F41F0F01 017A0F73 4F2377BE 7CFE753F

10500 - 172AF11B 0000E156 797AA02E A0DA4D11 9CE80D4A 1DA6F41B 15B52030 6A024868
10540 - 0D0AC2A4 E3312482 5A9678BE ACAA3CBC AB424938 0DFAA690 D0F0CA2E 55F17081
10580 - 2F2F2155 31C080D2 AA6A0D0C A2E57F17 9688FAF2 2074BE4F 0BF28E38 7E74AEAF
105C0 - 515F5774 EA38AF98 ED57E4E0 9B190B3C 6B30B388 E547EA3A 4538F124 B19B6928

10600 - FB34B1AB E8EE27EB F225A992 8A951701 5131C017 9A6F549B F7BF7AB0 A6F49215
10640 - 74170159 2178AB2B 36B46480 A3656FA3 A66DF227 9ED80A11 8135171D 1CDAE1A6
10680 - F4E615B6 2032100A 0CA0C480 A3656F80 125B045B 1D6AE230 2B96787C 8051796F
106C0 - BFD22131 02C6B045 AFAE0DE2 6735C805 8BD40C2A 0C5EE4DC D9804114 B648A860

10700 - 6A1B277A 4D7AEC7B 9C72EC7B 1D7F8C87 06063A08 70908D00 000AF01F 2B7F2151
10740 - 71E6E9F1 59717113 71F855F2 2B7736E6 E6145E6E 6E6E6174 2E7F1613 4AF22330
```

```
10780 - 32B15C51 650C56F1 362B7DF5 1F995F21 192A7DE5 DABF0BF0 2031A115 96176A6E
107C0 - 55F63837 D2CAF1AE 5F1F1F11 B6E9F2AF 2146B664 606C3F71 851031C9 184179A5

10800 - D405169A 6D43515B 815E8239 16902890 2AD9B2B1 9B6E0717 57C7565C F747566C
10840 - F7D55179 A5D51C7F 55169A6D 55F4F070 45179A5D 55F1F495 F2143D88 F674A11E
10880 - DE9F11B8 16ACA816 A4C580A4 E47415F6 2590A538 0D1AE28B F528B102 D0B24B24
108C0 - C40C5BF2 6C2A0E5A F8BB6060 5E17962B F74ABE31 19CB10B7 50B2676B A7BFAD57

10900 - 23B11913 412BD784 11C9A6D5 31851233 5F000026 22015F82 790A617D CAF0F013
10940 - 6E21366A CF79BAF0 F011B132 8A651132 E210B134 8616A5C7 132E28BF 40DB113E
10980 - A4868A83 611B1371 0BD68E85 BA11B137 10B87144 686F1F6E 9F214317 496C4001
109C0 - B6A822F4 F4832001 37C21375 BD238CD8 A1238CC7 A17A0AD5 1B495F21 46D7D224

10A00 - 30310B13 4A6D4751 5F82790A 9179E9F0 F0136C21 361796BD F72D9F0F 011B1328
10A40 - A611132C 210B1346 ADF132C2 8BB40DB1 13E24328 AAE112B1 3712B8EA 2C411B13
10A80 - 710B65AF 7C691C2A A0179A6E 49015125 2F73E911 1EA2AC21 451C4147 0C52F1E9
10AC0 - B5F31A1D 5147C214 5176A6D5 1F7A197B C872197B 491B8E9F 2D215631 62132CA1

10B00 - 32146B66 B6646068 1C1647A5 229A6D48 21521169 A6F48115 711799B6 0F9B2709
10B40 - 129D8537 DA876E87 2E876587 D9876D81 33D81317 AC813310 31317DB8 137DD135
10B80 - 841AF317 3D2CE133 12B8B695 12B13315 33932901 7962EF1C 31371351 3410A169
10BC0 - 136134E9 FA8E5DA4 11A135D2 30AE1123 EA123851 B676A9F2 97650118 B7B2C7A4

10C00 - 0B7B10B7 A88BF6BF 6F6F6F6C 6D5C6C6C 1347F9F2 C18E0E39 E1522DAC 91358F86
10C40 - 1B160806 FDABF30C 5AF031F9 95F21431 C4147EA4 FDD2314D EA540D0E 0532D331
10C80 - A0851E7E 1CA59F25 77BF11BB 7B10B8A9 B31B495F 2142131C 01331A85 5F1468F4
10CC0 - 01B11B85 5F2146DA C91448ED 1A911B8E 717FAFB8 E8A7FD51 B6E9F2A4 E80DF89C

10D00 - 94D2AEBB F7F7B615 70A69D1D AC6C6C2C 6F2F2A6E 80F01441 64CC41C1 5F915C91
10D40 - 691796CE FD214C8F 42010098 1610C808 08EA8CD7 F6C78506 570F6F6A F026A7A8
10D80 - 1C81CBFA 0D51F81C AE801145 1740C57F 0315B727 B0415970 115E7239 0A002730
10DC0 - 115C7018 6000D120 321088C4 AB078EFA F210CD12 032CFB8E 78B07AEB 8E212913

10E00 - 57C115A1 72D14325 90791249 1740144F 32CFB8EE 4B021440 2080FF20 34D44211
10E40 - 0B1B40DD 1790152D 11C8120A 8639184A 85B84D20 8E5C698E 07211FB7 4F22034D
10E80 - D0008FFA 0B18E9FD 311C8120 A8610133 A100288E 01581F47 6F2D2145 8EA12E8E
10EC0 - 292E1088 08908CFB 30840800 08EC4212 032BF38E 5D90571F 081C90CD 032BFB8E

10F00 - BF908E00 61BF7ACA 831808C2 E1E8C2F2 FAC214B9 68008FDF 67013713 4164C210
10F40 - B1122034 802E08A2 60618015 A5AF8165 1468AAA0 132C2132 06164D23 1D5DA152
10F80 - 4A4C4803 1F4EA136 CA8E38DD BF025A94 20102D23 0BAF532C FB8E75A0 07500134
10FC0 - DF135112 159AAC98 AF4811B1 35664F20 31EF74C9 93800170 AB815F7A F717215F

11000 - 21721574 B46AC717 3F2F2308 133135BF 0BF0BF0A B4101031 11AB8BF4 BF4BF413
11040 - 12032A00 B3193900 52AD015B 31371081 FE2FF179 204E17A4 6D2137D5 AC87A004
11080 - 70831000 3118135D 901137D5 1371577B 6645617F AC1D2941 3EB4515F 31738A6E
110C0 - E022F304 8FF0F301 3710B1FE 2FF17C10 431AF471 D5C24708 312011B1 3501AF81

11100 - 4FB66540 03291741 571911D1 179D215F 0C6C6133 CA131170 5EC17A15 B320028E
11140 - 92C25041 618E820E 8E71C258 21618EAD 0E782017 FAF2D610 B7910AF2 D610A03A
11180 - F056EAF0 10359E8E 88BA044F 08EB0CA4 006EA28C E1B61617 4727F008 7980D070
111C0 - 306B947D 8284914A 310F9620 0311F966 001618ED 8FD7F9F8 5910171A 21F698F2

11200 - 14313117 91534171 14784AA0 659185A4 31759482 45283100 5D1D2315 1C213514
11240 - 7D7119AC 694C6060 22A464CC A46447A4 64A417AD 015B38B6 F08AAA03 1636D331
11280 - 7315B37C 55D215F3 17317B14 577011B1 98F21461 340317E1 79AF0143 81CC6C6C
112C0 - 6DE8BE1B 1C5D255C 10986A81 D076C08E E133AF2D 9D710817 E179AF21 4781E129

11300 - D5DBD313 41198B7C 020DBDA6 0BF8EEF6 223B144A ECD45EA1 CE7E7D2E 60E6296A
11340 - 50B14D6C 32086AC0 8E4A6262 BFC6132C 264AF1F6 98F21411 31171147 D084A93A
```

```
11380 - E085A17A 17514B72 101F698F 21431311 4302C4D6 D78E5172 D5169156 416E16E8
113C0 - 7A90DB14 40314625 A80A8220 81E81CAE 0AE28A2D DD913416 A14E90AE 0077AF6A

11400 - 1D9068EA B6216F16 F167DB14 4077AD68 1D906038 FD55918E 80ED765D 1FF69F21
11440 - 49D2A6E8 B67096C0 08C6CAA1 36108AE8 78404E03 1926D416 5419681F 8E17E048
11480 - E1331B69 8F214013 11185211 611361B1 98F21441 34037BF3 59F13013 7C213518
114C0 - 448E1C41 43968DD9 64CE9289 17310D81 841C414B 96C4F50D F4F4F402 7FA35001

11500 - 30450174 1844E914 39681F92 CCE71DF9 643E1720 27B00ACB B46A4601 1F178F21
11540 - 51717F11 815D3173 AFB15541 72155303 75FA5031 3416F15A 31BB88F2 14013514
11580 - F16314C1 7214B188 1480333F 300021B0 4FCA42F1 63723250 168A031F 38C50AE7
115C0 - 66F777B1 1211BAF7 8F7A5101 B178F215 2716F163 1564AC71 621563AB 7703F4F0

11600 - 7E254808 C11D88EC 3E0102AF B10B8E51 58167D01 5A3713F4 98D590D6 06B7F1B5
11640 - 88F21564 AC716214 6D77A705 6062B48C 667D1191 351B178F 21567155 716F1691
11680 - 7F14615D 31741641 4E15D017 A143174D 2031F178 F2153710 017F1537 1018CA7C
116C0 - 0737E111 749E400D 5112A055 C07DDF32 6C609094 5A054B31 1B2490C7 090E2020

11700 - 4F1A0557 0D23088A E50B2610 B79C0481 33810002 E445F81C A74102C4 47EACBB4
11740 - 6A465817 07F90400 831606F3 E607011B D51F688F 2D214FCA 11AD68E9 A19D910A
11780 - D23102C2 8F4C4804 0011A10B 8EA2DD10 A7FBE180 1564AC51 84146F6D 5183D214
117C0 - EEA76204 001F588F 21574172 147AF703 8CB4788D 943B1100 1371C413 41378FFA

11800 - 0B194DD1 34412E08 A5F0310F 1C114D17 103A454A F1371351 28A45444 AF0A4541
11840 - 4DA81C10 01131631 58318323 AF2A9611 08E414D2 01583133 131120C4 D6AF0A7C
11880 - 8EEAB411 81350332 00E73205 00B3655F 01323086 21032408 690080FE 200B85B0
118C0 - B1F175F2 1431316B 10B3ABB8 BF4F4F45 E3137C21 37157417 015B5175 93C7D030

11900 - B84B0B77 BF5FA1C3 0B85B0B1 55217302 D0130719 F500DDB3 E45472F0 E8137061
11940 - 321318AC 11132CAD 6C913544 0E064101 3706C213 4DAC9135 7011565D 9D121733
11980 - FDD73A05 46E842D1 37C2134E 20680DF8 F7A2104D 3136DA79 D0184146 135E9DE8
119C0 - F401B113 3E007135 134186D6 AFF15C61 6002D507 D903D230 7C11F675 F2147134

11A00 - 18313517 21375C82 17F9E500 DDA324CC 7600E868 6FAF7F3F 3BF3DDAB 7ACF8130
11A40 - 17C7E490 7E6E5001 C6137135 D5134166 136CAE87 910027A2 07C9F500 DB135D21
11A80 - 45028F86 1B1261B6 75F28DA8 F80D2305 D5324080 37EFD4B1 77EF75CE 5A0D2145
11AC0 - 48E795C0 376957BD 95719680 2172D214 D72A563C F708F55D 7F6514D0 28F7A510

11B00 - 8C80A076 DC560633 7741A522 778B7141 17D91831 C2339300 038C3448 76FF4DE7
11B40 - EF04807B 4A037241 63007A4D 133D8133 1B178F21 5E62614B 9687217C 17715B69
11B80 - 12E017E1 7F17A5FD 2033E300 0320D413 1D1D231F 3B6543E1 4B968219 E0D0133C
11BC0 - A13154E1 33BF0BF0 AE4101D5 32308703 E471D231 F38E45DD 5F98C5E9 D75ACD81

11C00 - 19BF6BF6 133C0DF1 34EB1318 E5D89136 D5135D23 1F38FFA0 B1D91351 1114960B
11C40 - 01B878F2 AFB14C16 71544133 13116014 0137D713 517F8E1E 3F500135 172AF214
11C80 - FC33102C B26A0E20 021B178F 215C616A 15831641 42D23152 CA13115F 716515C7
11CC0 - 021B178F 21181547 16F11915 C51F939F 215F7165 15C70313 31011331 B178F215

11D00 - 6716A171 84AA4654 285A7F25 56067AD1 BC78F211 91351711 513172D0 15A38E71
11D40 - 3F4D0111 794369FD DA13216F 14615D31 8CD214ED 7AC9A4E1 73155418 114E1701
11D80 - 5501574A C7171130 16314E15 50170160 15A51595 D2305E3A FBA4EA4E 590328FF
11DC0 - CB81E175 14D87AF1 D23141C2 13414E13 61181361 6314C1B1 78F21711 5EA15DA8

11E00 - 6A911737 EECA07E3 61B178F2 1C317A16 F16515E7 15D717D8 7A6067B0 17A1B229
11E40 - F2142F41 6414A141 1CA94FF0 1CD14317 D6280A47 5417C48A 2470831D 662DEA47
11E80 - 590F0F04 A5A475B3 1A939FD0 14A161F0 F014AC4C 4C416114 E90AE211 91341683
11EC0 - 0115405B 1D2D01CD 15F31731 5B317974 09C4D6D3 6710D231 02CA1301 423150EE

11F00 - EB1451C9 14717F14 51CBAF21 5D5CE145 1118C044 F0A53FDA B1F8EEE4 F16114A3
11F40 - 13896601 77117121 8CC07F76 88504867 B78E065D 8E3E6617 1143D8CD 4FC17D31
```

```
11F80 - A2143966 90CDCD49 B8ED8986 3408E885 F5D18E9F 6F714E28 31505227 70D68707
11FC0 - F6B5D51B 588F2156 4B464A07 F4B65621 65350F0E 1115C5AF 231D2AF3 8F1C4804

12000 - BD8EC56F D215D317 3B2615D3 AF311113 17C1C4A0 7A1B6412 7E5C7B20 7D6A5FE7
12040 - 11B58E79 108EC54F 17211113 014E14D6 0FE1BF69 F214ED50 18E534F5 0096800D
12080 - 217214D7 46240013 37271130 10016114 693A7277 99110130 16C16F14 A304906C
120C0 - 01601467 67911013 0D231F3D 53230873 48038E8B 7F58114F 96A00133 7B01759F

12100 - 53E8E2A7 F5006439 8EC87F5C E14F96A4 E1337D6F 57E85066 008408E7 67F13313
12140 - 014E96A0 01611461 6A16B93A 43870D0D 2CE15434 428EF87F 16214A30 4906D014
12180 - 6F68E677 F18216E1 6B16A5BA 8E407F14 B9680017 C1471CC8 AA1117F1 7F17F17E
121C0 - 5DDD8137 D78E823F 570D0149 DFDA68BE 8ECB6F50 01331311 4F96A007 C00D231F

12200 - 3CA68EF1 B698F214 01301611 4693A001 6814E90A 000779C8 A1D90683 1A233040
12240 - 08CD6028 E146F400 D1B25C58 E027F570 14203027 66050017 114793AA 0CE15530
12280 - 38E426F1 37C21351 C4147843 96A72853 92EC0172 14B78DD8 E6E7F8E5 66F8733C
122C0 - 038E3D7F 8A4AE6E3 E8E4D5F5 00137CA1 311C4147 96E7F02D 8208E9A5 F53FD231

12300 - F314B968 009603B1 33CA1336 AEF7031E 6E606338 10002D22 03144E30 78F7A510
12340 - 20D23144 218F5A21 049C07D7 07134D23 1E3218FB 35101360 6DB061B0 F7F2D231
12380 - F28ED133 1C414313 0D214E14 5E4E4174 07145174 071457FA 0AC3A4FD 606D2306
123C0 - D5221B3A 5F280F02 18F58C80 07ACB812 15D57180 D174804C 47301145 D9067650

12400 - 078218FA 7030AC14 61B45831 117AD014 3D856CAF A7D707E1 006AF6A4 D0179B07
12440 - 31073607 40006008 D175101C E8D87510 218DDD41 0218D8A4 108FD979 0137C913
12480 - 514F96A0 01751471 34184146 132C2DD1 3616A136 03078F7A 51071401 C4147061
124C0 - C4147067 59F8FE3F 80739F1F A18F2143 DBDE1410 620D231E 38DCE510 1FE95F21

12500 - 431311C4 01D1726F 452D7070 6135D214 FDD8217C 1083131D BD558D07 E6E60600
12540 - 07E6E606 02041B8F 2E2AF015 A5A9815A 52591060 15A5A88A 05500B98 BF8020B7
12580 - AB475107 FBF20500 78157A10 6D250A20 A0690E70 8650085C 01749F10 077F3B72
125C0 - 109AFA76 34B7AAF6 AF773F37 C3297E50 A7E73F6B 7386050B F87A04A7 BA727A14

12600 - 119A7210 966B071A FBF6BF68 1E011117 404B7A70 63B7AAF7 769397E8 0AFB5727
12640 - D96AFFAF A7D737F8 6AFFAFA7 CC3B7311 9A7B97A9 38409428 0850BFA9 4880BF8B
12680 - FA745697 822AF225 30887050 A1E9FA50 AF097800 A9220309 B7250086 000BF802
126C0 - 0420313D 8E09F00E 2678B1BF 29288133 10001110 E26302B3 285C111A 72AF773B

12700 - 2AF52D32 0021B557 F2A9215E B91AE199 21199141 A920E795 A099750A 9718B81E
12740 - 90AFC097 2F281181 10E6D70F 27F3E75A 09F7639F 2E2B7A10 1B73AF1A FA7282B7
12780 - 275927E9 2B7272A2 111607FA F7AF6B73 76224129 7AC1AFAB 76AFF772 5AFB7A15
127C0 - AF6AF711 9A7B7FE1 AFB128B7 31B8F2E2 25152415 649429F3 200815E5 A1B15C71

12800 - 192003AF 22437B71 D2F29018 40500850 BFA02112 119101B7 EAF7501B FA7630BF
12840 - 967007B2 07D31A79 75517FC1 A7BB7970 D17A91A7 B7DA185C 6F4EAFA7 310A7AA7
12880 - 67554750 0B7101AF 22232807 03112119 101B7EAF 7AF167AF 7EFC312D 8E0AD0AA
128C0 - 80E2192D D00E2E15 02561734 D4807900 02AF67A0 179B0119 7E316521 04AF272F

12900 - 020312D8 C8FC0047 F306E9C8 16ACA103 7E8C1132 EA1AACEA C770DE70 A3AF497C
12940 - 40E4ACB8 12740061 7D20D1E5 1F917F2A 0E4C017B C554F023 1029E100 159B7DB0
12980 - FD0EF562 C01F367F 2AF215FB 80DB80FF 2B6F30AF 21F367F2 60701FD0 7F26B401
129C0 - FD07F26A 501F787F 2AF215F5 80D580FF 25A46AC2 500B9ABF A021F787 F215D501

12A00 - 1FB77F2A F215FB2B B06A0E61 DF1FB77F 215DB031 FF67F268 DF1FF67F 267EF1F1
12A40 - 67F214F0 31F167F2 14D038D2 23B1132B F0BF0BF0 BF0BF013 3BF0BF0B F0BF0BF0
12A80 - 104114BF 4BF4BF4B F4BF4133 BF4BF4BF 4BF4BF41 32031B10 9F211015 0716F111
12AC0 - 1507031B 109F2152 710016F1 52710120 02052032 00F92631 96860605 2AF0A7C0

12B00 - 2325159A E43AA2B3 2403AB0B F0A3E4D1 814A3E48 0AC053FA 44AC0500 B7403AF0
12B40 - 03AF0A7C 03203499 4008BAB1 FA8BE41A F2C44A03 200FAC60 3AF6AD90 3AFAAF22
```

```
12B80 - 4323A2AF 701007BC E7D1A70E F7D317B4 1AFA7DAE AF22C322 158E0F89 8E63B978
12BC0 - 8FA3EA3E 77BE8C84 6C00748E 713A7946 AF68EE18 0F0F0F0A BA7F5EAF 66ECF007

12C00 - 95E760A7 E1673467 E407C6EA F5D22031 018E274C AF415171 CF8C806C 007E1E7B
12C40 - C973E5AF 678E6701 02A31F22 431F265B F75A4BF2 BF225B96 30A22B96 30A28303
12C80 - 0D89000B F2B965FE 3082F01B 0F7A5072 C14D697A 70871A67 6CE9FE16 10211178
12CC0 - 10AFB771 0112A721 0A7F4B6F E08CF9FB 8C9DFB8C 4FFB8C6C FB8C81CB 8CC84C59

12D00 - 72F2AA0F 7CEF7451 45377304 E27C7B67 A01772FE 7A0F78CF 70314117 3104A073
12D40 - EA638020 8CFC19AF AAF22333 8CFA9FE0 0119A7A8 61E0BF8A 7AA7A471 7A8A5D09
12D80 - F2B0B7AB 7AA7A102 031072FE 0A0F785F 722241A7 ECD713FA FF111712 FAFBA791
12DC0 - 0A74DA8C EFFB7352 F1E11F76 DA5BE203 1B18CBD1 D80194AF 47DFE17F 84094850

12E00 - 8507BDCA F678DE79 785AC102 704C7233 76FB7A0C 7A7B7A9B 12A7CBB1 1A6B007A
12E40 - 1C748B7D C97E9EAF A700CAF6 86050A4E 676D74FB 76E2731C 1534B444 83747E2C
12E80 - AF234215 028E2A59 84084194 85085178 ACAFA793 CAF6693E 31A37930 400227AC
12EC0 - 0400AF72 27EB0400 AF522B04 7FA0400A FA7E83BF 2BF2A760 3AE78EE3 E8137C21

12F00 - 37203441 0008B600 85084184 281CD8AF 0A2CA2C1 C1CD4001 4B310296 29131B29
12F40 - 62D030D9 66318518 521C1CD4 B214BAEB 966C0AE0 A6C60108 F670B140 0BE0BF06
12F80 - 2DFBF4BF 40380FF8 0DFB045D 205A92A1 E040D4A0 9860055F 80DFAF2A 96BF40D5
12FC0 - AF030272 9A748171 BA1534B4 44B5721D 326009B6 0094C007 FEAAF1AB 88E11DB7

13000 - DA096C40 223030E0 2A0ED220 32563540 E69B1009 3900AB97 7C30331F 274BE400
13040 - 87200227 04F400AF 722743F4 00AF5249 0C9122B0 47E1F400 AFA79306 010B0479
13080 - 0F400AFA 718010A7 17210B7B 9204AF67 A6011297 60011B03 AF2AE697 270AF601
130C0 - 20330691 9EE80330 002AE6AF A0173708 EC25F704 1108AF67 242AF270 10BF2BF2

13100 - 1287B411 18AE6BF2 BF2AE9BF 2BF2AEB0 31FD07F2 20D23108 8E97F71B 8F2E227A
13140 - F230815C 7A7E8E04 228CE54F 047E491B 149F2091 448E244F 2031DF72 D4AF0421
13180 - 111AF234 000B4A7A 2F303746 01B149F2 1460A602 9A6E8C96 7FA4EACA 103739FA
131C0 - FA1FD07F 211B17BA 4E59FAF5 AF215FBB 7A2BA99A F77CEAAF BB79111A 7A11B8C0

13200 - 37F05AF2 2C342150 28EB1297 F29AFA70 C8AF660C AAFAAF22 13381516 2AAAF7AF
13240 - 22133815 1739AA7B 01769AAF A7B00AF7 74000403 21AF2306 696A0571 10A707A0
13280 - 0AF6A7B6 E5AA74AF 6A74A7AB F001AF9C 6C9D5C5F 5F2C901A F223A962 0972D033
132C0 - 6630BFA0 3C2C2AF5 C5BF2A79 BF2A79B7 2A75A75A 78BF5F5A 79F5F5A7 9D8F5F5B

13300 - 790305E5 20D23049 E9A0CC31 21C17C7F C3D23136 E3718FA7 B64B973B 97E607E6
13340 - F7A9057F 3136A79A F7AFE2BB F30C5AFD 7AF23510 06038ECA 9B7469AF 87F1FAF8
13380 - AFAAF2DB AFE2BBF7 0C5AFB73 A6D31219 ED00E4B6 101AF7AF ABF0AF22 032585B7
133C0 - A570AF00 1BF0BF0B F0365242 5636EF8B FAA7BAF5 AF220319 59F100CC 03AF7AF1

13400 - B65600F7 1E87C9F7 C9E78CF5 7F325639 BDB0E1AF 9E401209 6C402230 30E0290E
13440 - 40E5AF90 16602FD2 80F73804 31A7C150 71487901 5D673804 D0318F0E 6E56E796
13480 - 28566D20 7202FEE7 0F744056 C75504D0 31700E66 58B7B328 46751272 02706117
134C0 - F1331311 BE95F214 68B2ED6F E814A318 F966001B 9E6F2AF0 0115A535 FE106325

13500 - 916001B9 D6F214A2 00188127 1404C18E D6884131 7FAC3AC0 97850B47 7F817471
13540 - 0E26AD0F 079D0BD8 8C41388C DB78846A 4EA4E400 85601150 2201B7E6 F2146D53
13580 - 4E71E77E 301AE43E 7E30F6F5 1A301E72 30F5C5C5 C51AC43E 7E10F61A BD6F14AA
135C0 - 88A05309 1A101E67 0021F615 210E16B9 E0E150E1 E1501200 3D9890B0 2070605E

13600 - 379505E2 7250AA80 E2192972 78100287 6C150186 6C115229 4FB0BAE0 E26D60E2
13640 - E782FDB1 34037E00 0E26DB13 4BA801D0 CCAEA8A8 20348E6F 2136D747 018FFCD2
13680 - 31F38B6D 5C4D6C68 0F113680 91361522 C6C680D1 0D338421 200345B8 66BA4DA3
136C0 - 1FD9E6F9 5B18EF32 B8EC4B74 008A8008 310D8CA3 2B8CC8AB 8812746E 4617CCF5

13700 - 4E17F100 F48AC7D7 8BF40DD2 CE318FEA 43C30498 6BB34DF6 F2E2136D 7AF02E15
13740 - 2030A982 E0B0ABD4 B04B6486 69011815 C0DB134A F68CCCAB C8B1FF25 0F14A310
```

```
13780 - EAE53411 11196081 311EAE53 32222964 60D2E61B 9F6F2144 652D1DD1 F595FE15
137C0 - 27AF6319 E9620735 3B106097 20516515 A5315FD5 353B1024 9729235F E10C1972

13800 - 1135FE10 D3976112 17CDD314 F5F5217F CD314F54 516515A1 313D962C 25B31617
13840 - 7AE748E3 10FCC4D2 CC4626E8 EAAA1F45 70F316F6 31089A1F 2470F316 F21747D6
13880 - 65C00245 B1002554 100265D0 00275600 028AF205 BCE80FE8 C569B118 134D2303
138C0 - 8B9E1879 00CD4A07 3A166FF1 19D5D230 3E184631 FD8BCB08 790031FC 7771AE67

13900 - 071D6F6B B67561E0 590C0D88 56CD4011 C114F7A4 160FF8A8 32876E11 19D5D230
13940 - 3E131F68 BCEA31F7 57A70134 518AF400 331FE7A0 17F91037 A5413181 ED711013
13980 - 0DAF6BB6 77E0D671 E01C114F CC4A072D 060FF866 6075C075 B275A071 A07C4176
139C0 - 31031491 71151417 01512170 15151C33 07816ACA 7180A4C5 8F7FF360 72CC4007

13A00 - F2066FFD 07220814 81478108 1081003D 07A00F0F 07200031 4A16186A 00118812
13A40 - 812B6656 4AF91080 77360916 C7031FF6 31011013 0690014F 17114C16 186A0011
13A80 - 8812812B 664D0816 81610803 AF910807 74109113 60679400 71346F10 8F7A5108
13AC0 - C94AE1B6 98F21461 3403D906 136D5AF2 32002E1D D134128A F58C247E 79CF16A3

13B00 - 01154003 0C0C0C87 A6020021 18812812 A6E4C00D 8900052F 207F8FD2 3182132C
13B40 - 21341468 1E816816 CE812812 C6144077 55F81D90 60311013 07AAE23B 145A0207
13B80 - 78F02A1C D1A98203 010E0590 A50856D9 C61CF133 EA7B024C 007735F6 0C231F01
13BC0 - 4D171AF2 D9C615DD 17D133CA 131D4CC4 3114E1C1 14D734E6 DEF1CF86 660733E6

13C00 - 76011013 0D2308E3 56066D3A 8AA0C431 14E14D17 1750E6CE F1CF14B1 71153417
13C40 - 01532170 15351C31 51794811 2F309209 42508551 18136108 715E1691 56416A14
13C80 - 616E1631 1087AB0E A1406810 14681E81 0810AE6A 76144A4E A4E4A011 813403AF
13CC0 - 2111D6AF 014281C8 E6AFA201 118A940E 016B1401 18134119 8A670879 0003869B

13D00 - F1B698F2 14234430 00CA1301 421198A2 40038418 48740148 38E5D908 61908AA8
13D40 - 2028E296 71331FE9 5F21478A 2E01318E 3E904008 E035D14A 311F9620 0B669620
13D80 - 0037A647 6108E5A8 0ACBA464 0094A000 3D21F078 F2155001 1C21CF13 31F495F2
13DC0 - 147E2131 01171AF2 14717D13 3CA1F995 F2141131 0317117F 17F04133 62EF1B07

13E00 - 8F214E90 A81D2154 01FF69F2 14B8E636 D77D7088 58038C90 5E1FE95F 21471C41
13E40 - 431318A2 00018E3E BB8E023B 75938EE4 9BD2B568 B900AF98 E4B9B560 63551B08
13E80 - 8F215449 0800B04A 6496C11A F2A0E228 0FD411D9 13415678 E206B155 7021618E
13EC0 - 365D8ECF 2D75BE5C 0775F726 5D3788E4 6061A416 17F5F7B0 F6C00793 F41E088E

13F00 - 808056D7 B278AF60 69D0A47A 475606CB 0A474606 43187083 7FCC8656 06DE47BB
13F40 - 615378ED A6B87541 78A68E18 47779E66 9F650187 1F387281 1717C8C1 7F758C87
13F80 - 58B59B17 1704E1CF 7F6C17F8 65F98EA5 C08455B9 8EA38086 3801CF1C 176FD460
13FC0 - 6B048729 C782E77B 2873C969 5F870047 D0E748B5 2171DA52 31638CAC FB345104

14000 - 106348F5 F00685A8 CE7D673E D663F871 82872017 FBD7C3B4 AB57C719 D7E2B4CA
14040 - 71BD6CFE 8EA97087 3AD8723E 6A8F8702 37291590 11B688F3 0A982606 FA3748B5
14080 - D07E9C56 067946FA E871A287 29278514 8C30A98E FC171725 B17FD230 8E155A6A
140C0 - F017BD21 5F310B1C 978FC858 7B115B08 78686FB0 31FC902D 08783820 60A08787

14100 - 09EEDE78 2979F811 B8BAE07E E9315265 DED2303E 18468B07 0856D8C9 E35A077C
14140 - 968AE137 06816CE8 12AFAE0E 07C5C071 354A07F9 96262848 CD43114A 1C114970
14180 - B86DEF7D DA876711 19D58699 07D80460 673F8E48 A07E4B56 07570678 D7B26862
141C0 - 60621F86 3801CF1C 17EDB571 791C863A 073A065B E677E6ED 18C7A2D1 1013014A

14200 - 31FF9663 18696231 6310B855 0231FE96 24003869 A031D163 EF1198B9 6DD9E34F
14240 - C8ABACD4 CC4C08EF D7F64FF1 18136108 111D8619 FE3112D8 01D23080 10000000
14280 - 000001C1 31E014D0 31618ED8 1D8E62FC 7FDA5127 18B62590 8DA93908 31808C7F
142C0 - 2D61C08E 9F7F1681 5203010E 06908903 1D359181 69495194 101A454A 031C366F

14300 - C8E1F7F7 C5A56063 C0702384 5854D2E6 A4FA4F56 27B37443 31FF8696 031FE8E0
14340 - 27F8E3C7 F6670A4F 43386960 636FC676 075B086A 46658C31 FF8EBE6F 8EDD6F8E
```

```
14380 - 687F8C04 AA79D943 471A2A4F A4F54285 4D2E68AF 7087A318 69A08A79 08496C6F
143C0 - 8651C11B 6B2C8CC0 2B1618ED 9DA721E3 1F896690 31F014D7 90A60107 73A5606B

14400 - 8F087403 55F7722A 47A474E6 A4746062 E0862808 CD983D23 08870C27 D2651185
14440 - 5316386A 13523110 13015378 E765F699 F871C017 17889549 7834865B 8318110B
14480 - 601F8696 0623E870 C08E1BC3 60DF8718 1862EE7A 954398EC B4F634F8 72717CE3
144C0 - 8756B863 AC77BD53 C85774D3 74754DC8 4776C3AF 215E3163 C6132CA8 E674F69F

14500 - E8606063 80D23088 B991879A 01118BEA 031D16EC AC987A40 E6723545 A1101307
14540 - F2D8B9B1 111DCE1C C4F08EC1 5F64FF47 D15378E9 54F4218E DE3F8EE7 5F6C7E8E
14580 - 0A7F55F4 99871218 72611717 A58666E8 72836BCE 171AF014 381C7A24 45A8EC08
145C0 - FAFA81C8 E8E2F49B 8E627F6D AF8577BC 27DF342D 8477DB21 31D015A3 6ECF8C05

14600 - F48EDB4F 16115639 3A0085A1 33D88E89 2D4A0078 CE3ECD41 33D80313 71361F97
14640 - 6F214713 51758EA5 3684984A 84B14B31 2F962508 59136137 7E8F1671 564AC716
14680 - 11564AC5 A4EA4E55 085B1681 46D716ED 215E3109 16386AD2 AF015233 2EF10E36
146C0 - D681C814 814DACO1 00142D86 D00146D5 CB108165 AF2146E9 41181ED7 165146D5

14700 - 038C6E8F 70FE1537 30A98200 85085230 E9860184 2A0E9825 0853AE6B E690A008
14740 - 51878211 76D215F3 8AE00031 BCD6F2A0 65F3AF21 56430AA4 6440E61B AB8F2152
14780 - 2A2C421A 2C4C0DA8 E4BB6D61 7615D852 1872928F 7A5B0159 31731471 B178F214
147C0 - 416FD2CE 14402147 17615B30 B878200B 4CCE457C 72A01593 8101C3D2 8625415F

14800 - 310B1B17 8F214216 F164144E 6E6C6CA1 8FAF22E8 0FAA7ED2 15CA1845 E3FEF012
14840 - 30FB02DA 34A3841C 21341524 D68F485B 0201B088 F2150418 E142CA1C 28785014
14880 - 016F146E 6144031F 995F2143 13115341 76D015B3 CC017EDF 86781173 1438A8B0
148C0 - 131D015B 30315938 10D28623 41C315F3 1771438A C0113713 5134D25E 1130E6E6

14900 - C6C2145D 215E3E6E 6C6CA1CA 0330FB02 8F485B02 01731438 ACB01CAA F2564130
14940 - CA1411CA DA31019E E8529189 1567A922 0A044A3A FAAFFAFE 8EE0BAAF EAF7AFE2
14980 - 0AF58E92 4F5F0AF9 15578CF5 4F855031 6F15635A D9E69015 675EC312 18539E64
149C0 - 181C7B8F 162D0CC6 38F15677 6AF16F6F CF102D23 031118B6 606E2BD4 1228BD21

14A00 - C28B9D2E 9DAD2303 879FDCA1 19DDE04D 0E4E4E4C 952FC0CA D687A40E 68B733D6
14A40 - 6828171A F014381C 3010E029 0A70856E 4D2304C2 8B740038 7A00EBC6 AF1D586B
14A80 - A61F188F 21451F69 8F2143D2 31C1CA13 1147AE78 FF10A040 01378FEE E60E6E61
14AC0 - F688F214 58E7F4B1 43DED7DE E21C4143 D88B6005 918E45EA 400D6D71 F688F214

14B00 - 51F698F2 14313117 91534172 14313016 F16F142D 6C014017 717A17D1 43C01411
14B40 - 32101CAD BE2DF134 DAC0131D F8EAA561 321001F6 88F28F20 7A01B198 F28FE66A
14B80 - 01648FE6 6A0111A4 CA4C5051 30AF230D AF0142EA 16815E3C 68E4C0A2 08A940E4
14BC0 - D8183158 31B698F2 14234020 00CA130D 915C3731 A1537864 4003078C 318F1371

14C00 - 34135181 2B0C4907 A5046F18 01560F21 82156079 407F301B 995F2AF0 14213713
14C40 - 5EAAF88E 761F4808 CB77F31F 014D171A F4159D1C 16A2D181 14E96A00 1C114D02
14C80 - B125B1E3 FF1BA49F 2D215C00 11F08C41 72958F6B 02072611 B769F215 E70A10BB
14CC0 - F6BF6816 13586B92 86A60760 171C0869 6079F086 82C31827 0F067BF8 6852D210

14D00 - B14B3104 EA75B044 ED231D0D A7D6F029 4AF286AA 18695111 BD0AEAC4 E41036C4
14D40 - F79505BC 86A6C6D5 F14B31A0 9EE2334B AC4106D6 8F62420E 050000E4 000AF416
14D80 - 46543648 44D34310 29E25874 00601F11 BA6E4001 0B1BA69F 21461351 4B171137
14DC0 - 14431A09 624D31D0 962A07C0 064CF033 102DA1BB 74F21461 088FE3C1 01BB74F2

14E00 - 1421188A 6008CD4E 98F89810 8ECE6DC1 83105071 0B7C8411 B064457F 918F8981
14E40 - 08E113E8 F2C6005F 27481788 28EE17D8 635E3136 60C18609 A01110D1 AE84C08F
14E80 - EE010429 D43422DF 1C9C9134 8ED7E914 A1008E06 6DB1831D B31CC705 24606980
14EC0 - 31CC7742 1F998F2D 231C28E4 D1674A25 437F221B 998F214C 34B20008 BA40DACC

14F00 - CC4B3161 14F17114 C5CE1B99 8F23F55E 61637379 676E6154 716F3356 4615C37E
14F40 - F165DE31 0D7BB154 2310D76B 11103304 F57D914C 07A42216 9B0316C7 E8156131
```

```
14F80 - 6C798131 7F79B26B 00317F7F 6155472E 15E31711 533B24A2 4D617213 78E48D9A
14FC0 - A61C4133 81CAE61F 769F215D 78D08B10 34E40510 61108E5A A610F423 0452A078

15000 - 2B0C92C0 582A19E0 000731E7 9E625311 F74E07F6 1D0AEA80 FF203422 DF1C2C28
15040 - 12BF2BF2 E6638F1B 344F2D21 56292EB0 1A064F14 C6EAD8F5 9180D015 33177157
15080 - 40885B94 E5085A85 9B465128 491C4147 133CA133 1C114F90 A5085834 F9EE1C21
150C0 - 35E6BF2B F2BF209D 015B0E48 1CA8660F E8EA4E78 F127008E 44E7018D BA790331

15100 - 6A78C77F 58C145E8 C0F4ECEC ECECE173 1574AFA1 70A46BCA 80DF3502 B3A32002
15140 - 1F998F21 B374F214 E8168161 917D214E AF71378F CB9108F0 AB108410 B15438C1
15180 - B1B311F7 E7F7900D 58D8BC80 5E13107D 51108F1D 5405D0F9 794F540D 11188EA6
151C0 - B9A69013 13462A03 1446A903 13062903 1406A801 FE74F214 F1083180 8FC3C101

15200 - FE74F214 B1189620 08F0AB10 877268F3 90201611 4E96A051 FD32518D E0C10028
15240 - 0FF311F8 CFB3E317 F65FF8F0 AB108672 00A31E44 6031258D 1C32031B 464FF31F
15280 - 46CEF310 D6F00317 F72BF316 C8CF53E3 1CC65FF1 4B7F4E40 03102B6A 03315C77
152C0 - 4E357200 0746031F D76B0315 C7CBF288 F7FC001F 064F2D01 4BCCD231 E08EE699

15300 - 2034F64F 2E913515 723030E0 6A0C490A 2666FFA2 6441315C 7BDDAF2B 2660501B
15340 - DF3E21F3 44F21520 30790202 78DF1B06 4F215329 2C2C14AD 1AE80215 3292CFA7
15380 - 65C7A5D6 55F1B8F3 E22515C5 15A59126 015C5200 3CE24000 00000000 00000000
153C0 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

15400 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
15440 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
15480 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
154C0 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

15500 - 00000000 00000000 00000000 00000001 B679F215 641A675F 142D2132 15631651
15540 - 32CAA4E5 DE03048E 96808F69 2208F420 108F3442 0310C8E7 80E7EB37 4AF13167
15580 - 00776034 100001BB 88F21443 3000D145 1721331F B75F2141 174141E4 E4174141
155C0 - 1DE91431 C4141133 1C333110 F15D3133 1C414108 854031F6 79F21534 AC81E085

15600 - F7E62CFC 7D6134DF 1553E6E6 E61543D5 8161C414 B34D0000 96250AC1 17480DF0
15640 - D1C21573 93513A3D BB58E69B 5D25E2E5 4D0CDBB1 AB9809DF 134DF152 3AB8133E
15680 - A133A4D5 9B157313 3C222809 13416213 1DFEB8C0 DA51B769 F21523A6 C4E11481
156C0 - 62146136 D014A161 13614402 84AAB215 43A245D1 A243102A 24490318 2A24D0AE

15700 - A01A244A 031D06BE F0350631 A19E2067 212AE81B F88F2A82 15401A08 5F142131
15740 - 14B31929 65049620 187A6131 C2966D01 71137144 02033114 77C13151 D0AEA342
15780 - 76618CD1 9531C296 5019629C 3192966B CAE43D06 F5D5B504 B3A32D8E CD855B03
157C0 - 1B79E290 3112AE57 13048914 50231E59 610031F2 96100313 2961B031 E29E5000

15800 - 3AE53406 000AE71F B88F214F B669E700 AEFA661B 495F2142 EE18E142 E4E4E240
15840 - 0AEB14D1 40184131 CCCC8E7A 85AE4148 1371F085 F2031FB7 5F214313 1D3B671C
15880 - 114F96E4 F1C0037A BC8FFA60 03481000 DA4618E6 DA07A405 5F75FD5E E765E54F
158C0 - 771F5DE7 CA07C605 9E73345B D7FE45BD 8E64A078 9C60CF21 0D84D63C F8E215F8

15900 - EC43380D 23FFFFFF FFF10111 10F2094A D0B46400 85A02315 576008C0 ED096A00
15940 - 10971061 11D2AE62 88FCB390 71287516 11131149 62B03142 96600031 BB75F26E
15980 - 00D2021B 698F2142 1A085F14 6D518431 02690017 11331408 B80D1331 4B9629E8
159C0 - FB394031 0F966501 7134698F 2137145D 28730087 14487554 3178962F 83318287

15A00 - F35433AE 87294134 16114A31 0896201A E4789485 46C5F31F 4026851A C0AC1359
15A40 - 21F0F258 EB4654F3 87662319 29668085 585631F7 AE5AEC75 54AE4659 11BB75F2
15A80 - 14616414 463AF78E 74C170B4 73177AB3 71D48446 87000031 C7966517 5354F431
15AC0 - 12023165 0286BF08 62747BC4 6A503182 962C633A 63B73544 4D3308D8 764457CD

15B00 - 91341817 0A3872C1 312594D0 07AA3854 8558566F 5E311594 80070931 F698F214
15B40 - 7E6E6145 3182AEAA C9786331 9294ED07 E9C46131 C2A4E51F 8548466F 0E18917A
```

```
15B80 - 14714431 140232A8 29367787 7271B495 F2146135 D015B334 110F08A2 B47EAC17
15BC0 - 28FB3940 86B83862 60776434 B75F2137 1438A6D1 31D3AEA7 BC285484 58468576

15C00 - 68D32A82 ABA31989 62233019 62A2310F 962E4319 29625431 28966918 51307A8A
15C40 - 6D2068ED 31750233 08D87AE2 551311F9 666E31C2 AEA84684 5AB81BD8 8F214A96
15C80 - 8F078E35 01AE0148 74546310 85531450 2ACB7254 7B843142 AC740075 B346E185
15CC0 - 3311A8F1 F1AE5DD2 39157018 02415412 0861F033 11281831 5C375D57 D0616176

15D00 - 62854608 C782C1AB 75F14616 41428A6E 0164146E ACC01061 313192AE A31C2AE5
15D40 - 31D0AE71 C114F962 7F9612F9 63DE1711 33078632 513531B2 AE531D2A E717114F
15D80 - 96151967 2F17114F 967501C1 13714413 51648E5C 351371F5 85F21450 37EDF1F7
15DC0 - 69F2D215 53038EC7 731B2C7F 28FC1DB0 56131E01 4C161118 154716F1 50704867

15E00 - 0017F349 95F21371 4572D216 77A13442 76D274A4 8E289976 D474108E CB797531
15E40 - 02314264 FA1F2C7F 28FC1DB0 041B995F 21461354 411CF151 71371441 35021CF1
15E80 - 5171101C F151731E 01C114D1 53765DF1 B495F214 6011524A C8160152 416001AC
15EC0 - 0B44AC87 03070DFF A809FA13 61501201 801504AC C1801504 ACC13614 40333C7D

15F00 - 77D30340 69604D02 39060025 02742050 F313B962 EE31FE96 25E31A82 19660022
15F40 - 028C8315 1B585F21 4213116E 1421648E B0256F20 132703FD E6C101B9 95F21421
15F80 - 31184142 18E1468E 07151371 F495F214 5031311B B98F2158 51618ED6 59133442
15FC0 - 31A1AE71 4E983519 E301BE68 16ACAAC5 03AC0B44 AC8B4403 1BB98F21 585161AA

16000 - 014A3306 96743F41 1BE0BB0B 2416114A 1018E870 41524AC8 1BB98F21 5A50131D
16040 - 0AE5D3CF 17114B96 00031829 62CEE696 67EE752E 8D760303 70F82C2D 3278E710
16080 - 5500310F 96100319 29610031 C2961003 308D8799 E400AE98 0D03F778 11223455
160C0 - 56666AA7 AE680D03 F8892233 45666777 7B232001 73BD18E1 42134131 017DFD13

16100 - 68091341 51113780 91352035 D7C73B25 8E27F457 09400015 5417002A C27D6D15
16140 - A5319296 6C23182B 46400163 14A9661F 161744D1 5A59C290 9C100033 70F82C2D
16180 - 3278EF0F 45F03308 D87CAD40 0A4E7BCF 40031A71 4D17103A F4831741 FBF6F231
161C0 - 01966661 574AC531 20A4D482 31AF8E81 4D332000 28949928 F5C390AF 0A2C031E

16200 - 698F1431 EB75F141 8CEAD98F 5C3908F4 47B0AFA0 4031C015 74AC5311 0A4D41C1
16240 - 0131BF8E EA3D3310 00289496 18F5C390 11132105 ABA038F5 C390AF00 38218E4F
16280 - D4451AF1 A88815BF 503AB800 33110123 8E2FD45C E9680031 E19EE00A 0A66008D
162C0 - 1A0C59FA C103310F 14D17113 21B995F2 7A7C1C41 451C4145 1C414313 28E7B291

16300 - 318C83F8 30A982B0 31E0966F 08FBD670 8C3A298C E9918F34 420711C8 E11238E6
16340 - 0E18F13D B081CAD0 8F127908 F692208F E21206F0 C1B495F2 14218E14 61351410
16380 - 174EF184 146AF1D5 13518414 6E181D31 D0AEA14F 1B045F29 66013102 69001C11
163C0 - 4F7E42A6 D52F8747 16010344 95F21371 45757176 821F495F 21471357 A9049D34

16400 - 995F2137 1451D581 43132146 17E14534 084F2135 EA8F8187 0AFE8CC7 C4038211
16440 - 4A31D296 2C431B26 F3014A31 D2966231 61623030 79020D30 2902FD30 A9065017
16480 - 08F0E160 68008212 17181831 00171153 3171310F 962F831C 29628D35 82D30225
164C0 - 8E2DB458 C3192962 CB3308D8 766A5983 31496795 A51331D7 9622230C 962828F9

16500 - 705080DF 119135AF 66F7F14B 1717E106 17F14B17 1701075E 03364E42 3675F213
16540 - 06AEE906 00303F2F 214F1712 30234995 F2136D71 461358FC 1DB05427 0E012072
16580 - 3031C27A 801207F2 031826E7 07EB014E AE731E59 67619481 185AD831 92765004
165C0 - 1F585F21 47135D73 450000C3 14713506 8E82C107 133131DE EA81CAE8 A6D14F72

16600 - 10171A6D 52F86A00 31822113 6DAD7136 1A084F13 6E3D2809 9E7F0EE1 36154120
16640 - 02DB80D0 0D5AE130 2031E714 C020417F 137144DB 1348AA01 00030003 00030003
16680 - 00677063 506C8067 526B906F A0030003 00030063 816F4069 C0677161 C06B4303
166C0 - 006BC26F 336F2068 7168106F 237AA1AE 41AC74F6 B6079588 FD65007B 788EC61F

16700 - 74828E74 528F4201 08E6A6F1 B874F237 0800A4F5 15C70376 511BC74F 214AA6C9
16740 - E811031B C74F214A B6431A49 EEDB1481 61148185 330C0015 C3036192 8E9F1F8E
```

```
16780 - 7B1F5FE8 EC75F865 2E725931 286C1016 114D1711 321BD88F 214C1301 6114A962
167C0 - ED1817E6 9314240B 181390F8 2C2D3022 98E1B845 42161790 91361BF8 8F21522B

16800 - 24181150 31347CBA 74FA1618 EB47F76F E8448CC8 0F79EE1B 585F2146 18414431
16840 - D0AE5028 EF10F133 1F675F22 A1741410 C57F1791 431C4141 310C8E78 DC8F3442
16880 - 08D5C010 1BE74F2A E1A6D310 2161B651 4A9624F0 38E3DCE8 F344201B B75F2142
168C0 - 13018215 6393ECD1 321021A8 74FAF215 C708017D 2E1AB75F 14216414 68A253DA

16900 - CECE1441 351648EC 5841371B 585F2144 1AB88F14 EA6E14C0 38EF3FEA 6FDBA361
16940 - 33D81301 621F084F 28E02841 BF88F215 6431116C 0014D171 B67A4E53 FA6F42B7
16980 - 960D4131 8CCFBE7D 1F1B675F 21426290 112130D2 15631651 32CA1301 56393EA7
169C0 - 1B084F21 4C088548 5A714D16 5D3B6716 114E96E4 FA6F1B76 9F237000 084F2AEB

16A00 - 15C70375 AE1B675F 2142D811 28A03113 0182D215 63132EA1 30D21563 16210207
16A40 - 1FC74F2D 01411738 E2274348 D000EB8E B4648EF2 2E59F346 4A618CD2 646F4F6B
16A80 - 7C674F63 4F6F3F6B 7F6D0F67 FE6B6F6B 5C64AD64 4C6F4C37 BEE853DE 8F135104
16AC0 - 45843AF2 35FE10E2 AF515E59 75D2863B 316515E3 1631F1D8 F215D381 7310FD51

16B00 - 4E9EDD28 E3DCA561 77B18CA9 7D8F1311 05018F45 11031A34 0E797687 38085356
16B40 - 87C30728 510B5908 F1F39087 B0111B94 AF08CF18 D8DA9670 8DE9390A 97206536
16B80 - 0AF21F9E 8F215DBA F01D5C15 9B1DAD15 9E7CE68F D0F808FE C1108727 08614120
16BC0 - 33A30008 78246B43 77E351F1 B9C8F214 87836D41 31174143 19DC1583 728749C8

16C00 - 13DB10A7 E6611A23 736F08AF 81101017 D56B064F 29742411 0119EAF0 8AC3311A
16C40 - EBF28AAF 08178F45 1105A185 57DC4171 14976B57 5174D172 06A06471 A0F41190
16C80 - BC033C30 06E3F750 3DA1C315 F3719359 3860C033 73006CDF 76D58FE3 F8075C28
16CC0 - 75908D96 280D78F5 E510DB03 75F21451 F5D8F215 54875606 1A07AB64 EA789487

16D00 - A2873951 4476727A 85130706 5AF88FCB F3097041 76848F0B F905B715 071328F3
16D40 - 88608A69 287DC08F 45D70501 709584D8 5B72C585 A8F335A0 78257063 7F325831
16D80 - F9C8F214 97F55648 07BE4A06 5808594E E73858F8 44804B13 3B300D77 135DB6A0
16DC0 - E8497065 74B48177 3C1109AF 21DBE147 10A17414 710B8EFC 8A42C853 72C1D872

16E00 - 024606A0 11DBCD41 49AF01FA D8F2159E 8481BFD8 F2142E48 66B0CCCC 8A8D3140
16E40 - 878537B4 277B15D0 85294A84 5A5869E1 72041FAD 8F2147C2 145868DA 62607561
16E80 - 72928755 010B7894 77968477 96157884 27F41879 07338300 10871F38 721A8765
16EC0 - 58650585 684879C4 7DF01456 B4F74428 79F07032 876607FD 478A41C9 7FB477C1

16F00 - 70014B08 69E064BE 70E07B24 7E638FE3 F8074AD7 FF196803 86572871 627C5216
16F40 - F715015C 317314B1 64158067 007A2074 B114D96C 2F1D5C15 F3871003 3B400872
16F80 - 000372A1 7C938E5F 0B69531F 1D8F2D21 5F301707 24908673 4D2D507D 90184371
16FC0 - C2817AF2 10A8C44B A1F3A5F2 143E4131 143011F5 F8F21471 0B031F5C 8F215D38

17000 - 5101F2F6 1087CB03 20801F6D 8F27DE28 6750858A 4E118018 508A6508 40108101
17040 - 8E4C4BC3 118831F3 111D5AF2 8B070B46 DCE68A50 033412E8 A60034DD 67194AF0
17080 - A46A4634 E7871860 00037092 7780D231 46876703 2C218F5A 210AC240 0164D014
170C0 - 07C0FD60 601AC240 0B46D203 D88FD1F6 0CE1081F 678F2145 E18F956A 08CD90B7

17100 - 4F017514 38578BA0 084703D2 1FBC8F26 1108F264 101F9C8F 214B8148 14104114
17140 - 81081001 BF1BF1A8 8BF430A9 89903103 5603173A 61A4E5AD AF401D6E 6F2F6C60
17180 - B8615084 10B85840 08480370 9F7B70D4 13117C15 B5130038 138DA548 0131D817
171C0 - 81534172 AF21574A C717F84A 94A5114F A475C085 A1C21573 AF7780F1 F698F214

17200 - 313117F1 7415BA17 F1721470 3702F8EC C0B1331F 698F2141 58833930 0021038E
17240 - C24D113D 872BEE04 708BE2C3 36300028 D760508D C84108D1 84108659 08436600
17280 - 8538DB64 80208CDD 1B208CCC 1BDE1B68 8F214201 7C7084D1 188ED714 1D4E1472
172C0 - B8AEB01D FD1472F1 0A80FE2D 30420338 5002E8F5 C3901F6D 8F21470B 816A2E81

17300 - 285D5008 4D03DA71 EF0EF20B FC0EF60B 0EFA0B03 1F6D8F2D 286D50B2 6F20915D
17340 - 30371EF7 92F817A0 674006EA CACBA464 418EAC7A D24008D6 41A08EDB FA01010C
```

```
17380 - 0C80FF20 0E4233D3 0094E000 37D8D778 F747ED48 E8BF97B3 F7C1C131 1C414713
173C0 - 31311748 AA00C203 754E4001 7514587A 00CADC76 BD8FD1F6 0DAE8136 8DC33107

17400 - 2AB1B178 F2144C21 0B13487A 73161310 F8F9BA80 16113211 BEA8597F A07BDD1F
17440 - 178F2143 E2847037 CB040077 B04007D5 2D2AE67E A0400D23 027B4231 0F9669D7
17480 - CDD10B53 B781B7E7 04007622 145857D2 23D8B154 42767573 CC726040 070BD11B
174C0 - 86890CCC C1412380 92003000 00000000 00000000 0102792D 112C2145 8792187A

17500 - D0175147 C2145032 20CD2809 8591097A D4400136 5C084974 3D101736 D14410A7
17540 - F4D72DC4 A2C21118 79A387AF 2714CDE8 F7F3104B 0113797F 210D735B AFF7B0DD
17580 - FA4F018E C65C8A81 E76AC4CD 87AF0110 11A70115 1C1C9143 10175EC8 2284681E
175C0 - 83201856 D2E677B0 41A13111 0130113A F1D88660 1E587990 14A15908 4782281D

17600 - 83250857 879F1580 8E754CCD 57F4318E B04C1491 71CD51F8 6781869C 014A1590
17640 - 59014B15 8084B8E4 16C867A0 D0E4798E 86601733 CD6E4E27 4106CFE0 00000000
17680 - 0000000D 5C28BE21 1308F7A2 10400132 1F495F21 47C9145D DE1E8C28 D8031011
176C0 - B8F99210 EE134152 703E2100 6A10078A D780370C 87C62570 C01411FA F8F21554

17700 - 31D0DA7D 41400143 1C4141D0 1FBC6F21 5937B6B8 F7162079 998AE603 1B410A84
17740 - 7754B11A AC286500 71821011 FAF8F214 B3314009 0CB011AA 4E50010A 8F534101
17780 - B6C6F214 21C91411 317C7214 B3102966 5017176B A16B1331 36E27FCE AC24007D
177C0 - 9A111158 3799AD23 04EA1631 4835CFF3 0216115C 5310FDA7 46040011 AAC2B460

17800 - 2869811F AD8F2AF0 1431DBE1 41032BAF 0A0CD286 65080910 B7420400 7F514007
17840 - 75987A90 8FC1C700 3D0CC220 C100D280 97791AC2 40011015 07034010 046EFF7D
17880 - 7B776140 08775C7B D9D2304C 21451C41 4511B135 1347C218 FE5F4084 7AC24007
178C0 - AA913214 11357241 100AF015 B9AF2390 21202F30 29765017 71337769 E2137177

17900 - 137708D4 002380FF 11071387 4491587A F0704981 C7848511 101D07E1 F4001117
17940 - 1E01C414 71371C31 59313714 5AF2310F D57009DE 8B986B46 33140002 869E0758
17980 - E81C1410 38657487 624D230C 10B7A78D 2145875E 087A9074 B8103727 B4007958
179C0 - 87A0111B CA1CF1C2 141038E4 88FD015A 31B4E8F2 140011B5 F8F2144A C201208F

17A00 - 5A210400 130C0141 031B178F 2D3CF8D0 E920D6F0 F0F6F6AE A03959BE 8FB9E902
17A40 - 031FF8E1 6F846068 D0AB8762 1102D410 01711371 098F9AF9 0D256031 13110C28
17A80 - F5A21056 06031119 134AC215 44D6110D 81128BE8 1134C913 5E9E28EE 46361301
17AC0 - E995F147 130C08BA D1134E91 35E0100E E8E09631 1013020A F237B3DD DDDE1547

17B00 - D91371CF 13613717 F8E89538 408538E2 F6862928 D38090E5 8BE8FB9E 902031EF
17B40 - 8E66E850 EDC70302 59096067 60D9D713 01311771 37208FFF E6013613 5DB659F2
17B80 - 01C81793 2A00B314 E214F9E7 629E37E1 5F7AF5BF 5AE18E85 88F0F0DC 0165C320
17BC0 - 8D954A03 C8DEB66B E14A3132 966808CC D2C13610 88F88B70 1181348E 052C8EF1

17C00 - 2C8E40BC 84586250 855861C0 8EACBC67 0077231B 765F2146 D71A296F 1468AEB0
17C40 - 1A265F14 613414A3 1C2962F1 316C1619 0C501638 F17A8031 02595161 13210213
17C80 - 18FC2B90 1038F9DF 30D81338 E4B90109 131D4311 F962D030 09669517 31C18759
17CC0 - 5860E020 31F18C2F 288631F8 FBEC2013 71F296F2 14511913 58E3DE01 37135134

17D00 - 3497D718 C45B0850 865EA570 863CB11A 13411913 511BD714 A31C2962 7116131D
17D40 - 0962B0D6 78B751E1 361B296F 21448507 4548E04E 08E2E807 BB713758 01611368
17D80 - E63E08E7 A0C41108 8E679C56 06E6E8EA F4914A16 1311F966 60603E8C CC5C8740
17DC0 - 01FC78F2 AF015B08 1C170137 D77323D2 5321371F E95F2AF0 143D7EA8 1C730314

17E00 - FA6214D1 F278F214 706038A8 00D8137D 77CD2171 D014B968 359EA71E A8B874E8
17E40 - 8749396A 43C87640 DB2E8F36 B9010A7F 5F11A2E8 FE4B9013 58E8024D A5B97610
17E80 - 570D4AF2 10A7B6F1 1A6EDFD9 8CC6B3A5 000D8112 D413677E FDB135CC 44310214
17EC0 - B1711378 6CC08FDA 67059110 98FE3C10 11913511 26CCF8EC 81413403 864008DF

17F00 - DC10C45D E096BE14 A2031329 629031F1 5618CC63 C725DE67 6BE31F07 DF514A31
17F40 - DF966F08 CDF438DB F1908447 2FF8F0E3 202F3101 FD107F08 00066118 5416159D
```

```
17F80 - 8C17C977 71DA3151 D58B470C 158F1711 4FCE4709 E1D1E07B 9631027A 45E459F7
17FC0 - 02E63BF7 6FD6BAF1 3610B330 3008F3C3 9011B134 D0642016 173358FC 1DB042D9

18000 - 4CDC8E91 23CC42C1 007C2577 E0110D51 7114F96E 50B268B2 70EA58FE 01005B0C
18040 - 0100797D 8F818701 FB98F215 5734B98F 2D7D2308 110EA4F0 100DA718 D60EFCA7
18080 - 77D84460 DE310F9E 292752D3 484A8006 1F278F27 DA51C414 7133C206 037A6470
180C0 - 8414BB04 A6496831 1367E301 348517B6 08E7A238 44171AF0 14381C17 D781D695

18100 - E1F778F2 147135D2 14F038FC 1DB0041B 2C7F25D1 16F1612E 15071C81 8815370C
18140 - 5FE15070 18418506 C0081184 08417335 8FC1DB00 44E17170 7B704709 4C607180
18180 - 7A806E20 75507170 31827C63 727031C2 70631107 36031927 1537A401 19133131
181C0 - E2BF2BF2 30F1CF75 3315D687 0008C521 E17F1371 35109010 4B24A2C5 0096C000

18200 - 38610031 0269F204 10278DFA F25B135E 416E47BD 2BF6BF68 AE3F0194 8A031D27
18240 - 1C2B24A2 C3594E66 643D1331 FCD6F214 F0B84284 30B13311 290E1695 CA031036
18280 - 48205322 10D5220C CD9089F0 C80FF04B CE0520B3 85813200 59B64232 1109B642
182C0 - D12F0CA3 D5AF80F1 20309441 322109BD 2E812F23 0204BF3B F3BF3AE7 2130C98F

18300 - 50A87AF1 ADCAA6AC 005A2655 0B44AEB0 B8716061 C00B0421 30DB0B80 D105AF2A
18340 - 99A71949 E0BF5B34 550AC0A9 104A1D20 A8105948 F0BB8329 94932A5A EB0B8702
18380 - 00B5B432 003AB7D6 B3B5CFA3 BBB793F1 F93AB2A8 5EA948A0 303CAB0D 0430DB09
183C0 - 80D00CB0 54AFA0D0 4E42EB90 908AFA1C 6F90674F 8400BAE7 80D1D6E6 0D5BFA26

18400 - 43E0D891 63A3E55F AF295993 A99A79A9 297A0C94 A62BF6B3 4550AC00 D8813120
18440 - 32110229 B6A9AF9A F5948D0B F50DB345 7F04A1D2 0A88BE49 088030AA 01AF0A7C
18480 - 20A84A07 BF7BF7BF 7BF15F19 49D230EB 0280D030 F90560D0 CC20AC9B 454F1303
184C0 - 8127D30B F1A0C56E 31E2BF54 BEB44400 31547C10 A4C948A0 31D27C00 AF8AF0A7

18500 - C45C1C11 378B3711 3714D03C A7031E28 BF006B5A 8CEC7A8C BAE231F0 1F078F21
18540 - 4D136061 B078F214 E90A63BB 61542A0E 50339D49 F24D0A0A FA343AE7 11B778F2
18580 - 158B1841 44071340 3A0E5138 EA9D9F08 31B0343A E71DA1B5 59F215EB 34659F2A
185C0 - FE66BF8E B6D9E083 19B5F800 84013606 13313176 A01BC45F 215651A0 84F7A204

18600 - 611A045F 15651A0E 4F731086 0A031D07 1EE07636 031F2AF5 14E16196 A00A5546
18640 - 07FBEA6D 57E03724 01F995F2 14713503 840136D5 13371201 3114AD6B 644C0738
18680 - E1615DED 91346A2B 1018CBCD 1CADCEC5 BAE79741 33130103 5F316114 A9667F16
186C0 - 114A312F 96292775 E8F13DB0 1C2E4E41 51317213 772134D0 7E133402 B817F138

18700 - F456108F DA670460 6BE37538 8EB75C46 06F348EC 633E0F83 01BD321C 5331D634
18740 - 12735192 39142381 9F30071E 28F15710 D23037EA DD68E764 03260015 537A347E
18780 - 935B08E1 22217131 FB965606 4541378F 83AF07F7 DAC98558 0DF89CA0 89D50845
187C0 - 207C7413 61B198F2 144775D7 41279008 75036AE0 1B495F21 4613410B 7CB3D755

18800 - 01318478 D3EF306B 518739F3 11F962A0 3009668E 8FBEC208 F2713077 1EEA1513
18840 - 76831371 341351C4 11114134 C6881063 48F5F006 62797EBC 143137C2 134164D0
18880 - 1523E4E4 1503136C A1B995F2 1441311C 11A198F1 4613414A 161311F9 66351483
188C0 - 1C2966E2 13662DE8 70B28636 2311F962 A0300966 516D3F8C 49AB3382 004428F2

18900 - 71303362 0054114B 31D09627 D3372001 091B995F 21421331 75102133 1A6C6F14
18940 - 07BD1137 4903451B 81DA11A1 351192A8 F1F39068 9D7FDC7D 70131134 1098F460
18980 - 30136137 136137D5 13514A16 131C2962 D131D096 241D6735 B3102966 3D58D136
189C0 - 1B995F21 42130E21 543D9135 8507DAC7 6C1774B7 40065AED 01533137 CA135021

18A00 - 0B7A1113 712B4001 2B3451B8 112B011F 6C6F2145 11BAF0DA 1038D656 901B045F
18A40 - 214E0A84 00B14C8E 387C1A04 5F0914C8 F6922003 8D1D6108 F4561070 CF8F3961
18A80 - 01721377 47F7F8F6 67C71DFB 044BD986 6D15905F C7BBFCC6 DEF71BFD A58E7191
18AC0 - 7D8B8507 01B8F83D B03102D5 CCCC8A8D 117114F9 61EE8EDE 6C8ED77C 6B1C7A3F

18B00 - 8E481C85 E6A40792 F6CDD22F 30222000 1F976F27 D2B17114 BD817114 B17114F9
18B40 - 68001C10 37DEE8FD 65008FC2 6908C310 E1371097 FCE113D2 30770A91 35129134
```

```
18B80 - 17211BE6 15D21728 E2E521C1 31D014D1 191356BC B7B00153 78D8F5F0 1371F995
18BC0 - F2145135 0113211B 130EE109 8CF25276 8CE416AE 6ABA1371 08775AAF 21701473

18C00 - 1F01CA7D F815578E A7728E43 81171850 786A1188 F83AF076 3C7C0074 F87B6F6F
18C40 - AC133131 71F21310 18D01500 B38CE04B AE72C88F 13DB0AF6 AF0CE431 2080F08A
18C80 - A402F153 11F2B7F2 1517695C 200431DC 8E6A9A5C 11F2B7F2 153797CE 031DC8EE
18CC0 - 39A028E6 6248EDF1 41F85D81 8E855C8E 5BFB5B03 1D096221 7D5F724D 8EA32403

18D00 - 1F995F21 47135850 77C88F13 DB0D6CEC ECE4CC80 F08AA402 FAF01531 1B2B7F21
18D40 - 56720976 8A7CEC8E 4E14646F B1C30716 373777F6 2746F302 FF8FACEB 3AAE8DF3
18D80 - 6701FB49 F201078E DE901441 3416385A 601081BC E3FAAE84 A14A310F 9E280AF0
18DC0 - 5618F67B 90460685 3B4758FA F88F6B5F 078771CF 8E8CFA46 064748EF 968AF415

18E00 - 17316D8E EE7A737F D0149744 71CF1371 C4145137 14A8F504 50402161 313F9664
18E40 - 155114A3 18F96690 1616E611 36061361 4A313296 6C31618E AB5876D6 D6BF2155
18E80 - 7AE81361 088EC168 5E011813 48504118 C6C588E5 C1184071 DE14B14E 16187070
18EC0 - 90A6063A 0D2B568B 537B6414 9077F662 E31029C1 80A81543 310D9C5B 2A811719

18F00 - 418017F5 A117415F 828A9D1C 48E6E421 CFAF9155 71C020A8 215D0730 66A0F8D0
18F40 - B7F07606 71FFAF98 F928F007 5606A038 E833866A EB641490 730ADA74 D514F986
18F80 - 0381690A 808CA3DE B465C117 1143D231 048BA808 C171B1C1 1C030186 050B0662
18FC0 - 7F136108 8F17B704 E18F6277 05D03340 2E8A6908 F39B70D2 8FEA2105 606B628E

19000 - C13D1B99 5F21F178 F2736513 51C0D286 A5030887 D40E6155 08E5F01D 214C1BDA
19040 - 5F22C8EA F011840C 54F1BD86 F27A0571 151B765F 27BF41C5 3500F40A 15D57CB4
19080 - 8E1B2973 5606A767 AC41CF15 371028EE 3804438F 4CEF03B0 23414C4C 40215CB1
190C0 - 6B11A97A 90154716 F8FCBEF0 11013014 A312F966 641618ED E685606F 068F36F9

19100 - 044F1338 E05401CF 157710A8 438558E1 21153131 136F438E 680141F8 7341310F
19140 - 14A9EE80 16154F31 6D8E7BFB 1F995F21 431742C1 411740C5 7F3191BF 0BF01FEB
19180 - 5F215961 76A6E55F 7FEB14F1 0913610A 324088EF 0784918E 7E888E3C 78592D21
191C0 - 454DD137 C21351C4 14711196 8F08EC88 84C16D70 8EAF0917 1147B364 3E155311

19200 - A1348636 066F21F1 78F215F9 1F198F21 5D97C5B1 4B318FA6 C52114A9 62606332
19240 - 6D321491 4A962FE1 61D28FEA 21050179 1578258C D08014A3 13296673 1617ED28
19280 - EBED08FE 8111D2A6 E8BA0171 E470F48C 8AA9AF88 534D18EF 194768C8 43D2B568
192C0 - B5606BA1 13606739 310A8636 06461890 6061A08F BB2B011A 13515372 F30C942A

19300 - 0A469469 39444720 8108101F AB8F2157 0906C5A4 631CE94A 5030FAEA 15076980
19340 - 30C9C6E1 94D53D6B F231CFB4 28121547 6560B469 46ED9441 1BF0BF0A 6C4DB584
19380 - 6FE08929 F20A82A0 E906D381 6A4EA4E9 45ED8F75 8F011A13 52F30DAC 52015378
193C0 - F3B6F007 13420655 E94DCA8F BB2B011A 13530A15 37986901 50752DAF 230E1093

19400 - 1021088F E34B0D41 3016A146 132EA130 11A13517 1153717F 15771547 16F53B88
19440 - 2F220BB4 D6F2B06F 2F2AE910 A8EE0684 606BFD11 A145695F 20741331 4266251F
19480 - 198F2143 1BE95F21 46EED717 4147135E 28AAC5FA DF1327BA 431B5AED 8F725B01
194C0 - FDA5F214 7EB14513 57EF1133 D23133CA 1312C147 CB145174 0C52F171 147CB145

19500 - 1B265F21 4613485D 1F386F2A F215D917 915D98EF 5E086860 692170B3 6CF01371
19540 - F995F214 5135011F 995F2147 137011FE 95F260FF 1C915E91 5D91891C 915E915D
19580 - 90115F91 5C916917 915F915C 917901A0 0CE00000 8DB46708 FFD09077 234728F4
195C0 - CEF03D02 54E44435 552415CD 16D8FCBE F074A14C 28FFF980 100119A0 640690E9

19600 - 28FB5190 4A079D26 F108F92D 708E4CA8 3193655E 8F311901 1013031C F14A9664
19640 - 131D0161 14A9667F 1618D7E4 70161136 06031FB4 9F214FAE 71F198F2 14713520
19680 - 14F170A6 F4C390E8 017F5BEB E6B06561 17114717 D133CA13 15ECB065 8017117F
196C0 - 17F5CB80 D0153713 70314317 43400F40 8A200F4F 4137CA13 151E7C60 5008E63C

19700 - 8635808C 8ABA7450 1317CBF1 751471C4 8AA001BD 55F2735E 7D40795E 169744E1
19740 - BE95F22C 8E02A016 40C54F8E AF9014F1 09031F2B 5F21471C 41438A60 0038CAEA
```

```
19780 - 81B976F2 017DEF72 6F4001FD A5F21431 311CF1C2 BF0BF01B EB5F2203 19115861
197C0 - 66A6E55F 31B5AE78 401331F8 A5F21471 318BE606 4F014B31 F50E6296 3606E708

19800 - 60E01187 D468B2C6 14EB6614 C1611371 35144181 17214B31 A0982B3E 69EA4331
19840 - 0E9EEB2A BF1088FB 16B0EE7E F5C2128A B7870501 021CA850 1C51CF6B 5F860C11
19880 - 2A135147 133EA131 1C28405C D14B31F5 0E663114 A6F1869E B4003161 17F17213
198C0 - 71441371 CF1C2181 AEB966EC 662FD055 C8D81EF0 8DF43707 29BEF979 E16414A9

19900 - 6C25D118 4146E649 47B45182 31AB14A1 61962F13 19B966BD 16414A18 496CDCB6
19940 - 557CA6D5 1C1838D8 5A8062B0 6F4C8CBA 7177FB13 21B7B5F2 14EA6E46 2AE71611
19980 - 46134A6F 4B016216 F53F1563 931C0311 28C923E8 D765F0CC CCCCCCCC CCCC0E13
199C0 - 21307DB6 AE21560B 065A31FB 88F2CCCC 1417B7FE 4860D08F 7A3F06A0 08F182F0

19A00 - 8FDE5F08 CB79AA0E 1601321B AA8F2108 15401841 40184133 140D4101 316D8EA0
19A40 - C95A0311 5675F8F8 8B707056 460614F1 F0A8F214 71358EE3 3A4808C9 05E31080
19A80 - E6973868 6004312F 14A96643 16171D51 6117F8E8 7714808C EE7F2490 C202040F
19AC0 - 17515931 C5160313 F14A9665 216175A5 8EAC2A5E 87D16319 214A9665 E161345B

19B00 - 8F213614 41841371 4410A18F 14213113 48401563 A665606F C0162128 A0E4A212
19B40 - 8153730A 982F1171 31E0962A 2D6B06A6 696A356F 09156393 E5F15071 7F16F66A
19B80 - F8501469 3ECD1483 10278E51 56716F15 27154718 F1507D23 10268301 469069AD
19BC0 - 2304143C 216281C1 58318275 A5142D61 6D81C159 31738E58 51D91361 35692F14

19C00 - A31F70E6 61481621 4693EC18 708016F5 6431E014 C3102651 0162D215 E3E6E6C6
19C40 - 1827835A F215D386 0D015571 7F1557D9 13613512 8A0E4606 6FE12213 7135D713
19C80 - 6134E3E2 70DC78E1 16814E96 E1172D15 A031F46A FCD2318C 74A1EA1B 495F2146
19CC0 - 8B6606FA D1C0D287 D40E6155 01BF69F2 1C114E14 D1BEB6F2 1461C215 537E241B

19D00 - 3A5F22D7 A341840C 56F1B188 F21CF156 7155718F 1CF15671 5571B698 F27C412D
19D40 - 077A410C 57F1847B 317B2A1C 47C181B5 A8F27221 D21451C0 CE15501B 995F2133
19D80 - 2D140164 0C57F1BA A8F2D215 60E61F7B 5F214D17 1165146C B14572B0 4B18F95E
19DC0 - F072B08F 078708F9 AEF08E03 78D37890 1327A991 40D2309C A1FB88F2 1411B5B8

19E00 - F2146134 108311F1 4A966D31 19AB5310 20E6596A 11743B8F 7A3F06E0 0752B8F1
19E40 - 82F085D8 FDE5F06B 608C7B7F 13313101 132CA130 01737A40 08DE2C10 1BBA8F21
19E80 - 46134183 011C4146 1451C401 17414768 D2627BE0 00001F07 8F230115 D08EA468
19EC0 - E370B156 063DAAE5 769F4018 F95EF074 91AE58F5 73B01471 0A7B8817 48E296F1

19F00 - B198F271 9F2D1470 61740C55 F1647B7F 1B178F21 57715471 7F16F157 7154717F
19F40 - 1B995F22 D7122164 0C56F7DF 11BEB6F2 14715431 721BF69F 214F14C1 7111A157
19F80 - 410ADB13 5172A6D4 8017F53F 30A15379 8297AB81 7A14777A EEE13417 4D2B0544
19FC0 - 316F16F3 1027D891 57717F15 3715571C F15171C1 31E014D6 B2015E31 63C6D57E

1A000 - 5E76591C FAF2D9BF 2BF2A0E1 55713710 B8FFD090 112948E1 7D3E4018 F27C808F
1A040 - 3119084D 6B107A1E 4F011A13 48FF8EF0 74881121 3011B135 8C092C1F 7B5F214F
1A080 - A6E018FF C4F08403 1020E659 6A008500 28F88B70 1B265F21 42130184 146D71FB
1A0C0 - A8F2E754 003CF136 CB134146 D718114A 16131989 66AD1321 30141166 798F1199

1A100 - 31001305 CBAB9155 3172AF28 609034F0 00215571 C2011B9B 5F27E001 811C114E
1A140 - 14D031C4 146790DE 2145031B 7B5F214F 14C17116 1146D714 773ECC21 44174011
1A180 - 6ADA1371 2AE21358 E12C9460 6CD81361 34E21441 6412A102 136D5131 01843845
1A1C0 - 76C111A1 135F597E 9085363F 07F11495 13613413 51C4786C C01C3D21 4F133C21

1A200 - C1E5590D ADB401D5 8FAEB70D 41371F26 5F28D4DB 7097EA03 1F3627A1 F855F214
1A240 - 38447C41 4A587500 DBDA1351 4F96E7E8 74D18548 FEDF014A 11333038 169C77C8
1A280 - FF20115A E873001F B49F214F 96E00853 56911387 3117D404 B97F2069 2F131157
1A2C0 - 71129768 E7810112 1038F08B 7013011B 10A031FD 55F21131 4117EDB1 45013452

1A300 - 000CA130 1468AE53 87800D68 F2910144 00213213 111A1088 F13C7011 311810A6
1A340 - CBFE6400 CE721B14 6D518114 A311C168 966118F7 E77011A9 7231E540 0D971EA1
```

```
1A380 - 875BC181 031FD55F 21431031 3117F17F 14771BAC 2D71CFD0 15B38583 4402E08A
1A3C0 - 20084834 412E08A2 00038C8C EE690BE5 A39E14A3 10F9E261 1FA59F23 44D0A014

1A400 - 55848EEE 888F83DB 0AF23068 B6AB1BA5 9F215001 60137C21 3581CA0C 4111C114
1A440 - F14C1615 DE61C28F 83DB0137 135C2109 1371F495 F2D7147D F1350113 71F995F2
1A480 - 69EFF00B E4139E8E 468876BF 31D0217B B0D23021 08328087 880D8310 C8BA958C
1A4C0 - A5C9FCFA E4D29E8E 4288767F AF2297C7 0D230B10 832BFB79 40340060 08B240DA

1A500 - 32C00D1A 31B3A59F B31D4101 11A8FD79 11521110 1301198D 78B808C9 4AD10A8F
1A540 - 13DB0137 1288B600 11A8E1B9 260FE840 137FA809 FA8B38C1 37155120 6577DB2B
1A580 - EAC69E13 61088FA7 7708FF8B 70851852 7FA14C11 F976F211 A1458F7E 6908C92F
1A5C0 - 01101307 2C3102D2 E6F21081 09F2F2F2 10B14A78 21436100 7E114951 0174114F

1A600 - 4D613210 28F4EFF0 8606066F 013712A1 347CE047 20523B14 2004491D 68F4EFF0
1A640 - D015B387 0501037B 33137112 8A232130 1838F788 70D215F3 1108B680 8C42CE1F
1A680 - C65F2147 D711A135 D215F313 411A1351 11D81378 BF85135D 015B311B 8BE74110
1A6C0 - 8B241041 36108D2E 610954C1 59305C00 4100173D 214F133C A13114B1 719087A5

1A700 - 9E851842 77408C67 C914A161 311F9660 0D015A31 63031FD5 5F2143D2 3152CA13
1A740 - 11438A80 0AF215D9 841041BC 65F2146C E8F29101 4C031C38 CB55D741 26E91137
1A780 - 10A13517 184314B1 718EFF21 CDD21DD8 21ED3210 E120DC35 05CE40FD F605F170
1A7C0 - FEF70005 B514B313 E96231E6 96261853 78C14501 7117169D 017174B1 17154E17

1A800 - 314B31DF 96681171 14B17131 F0966606 41164607 18117114 B314F966 5B1716F3
1A840 - FD015B31 73311096 27334FF5 108A2D8A 2E8A272A 2E8A2F18 61618E2B 59935606
1A880 - 5BC83114 647014B3 12E966D1 17114B31 4C962B03 15896650 17175F03 14F9662F
1A8C0 - 1716D3F1 4B17131F 09620631 4C14B966 5017173C 08631114 B171311F 962CC112

1A900 - 131D214F CA13114B 17190CA1 1731BC65 F2142137 1358BA00 6C4ED287 1B014517
1A940 - 857A1371 35134862 42174D21 5F3D57C2 08F23970 50013613 54CC143C A13015A3
1A980 - 17415931 3553B1FC 65F2147D 71CE143D 23113CA1 310184AD 014B3122 9EEB2171
1A9C0 - 303A02B8 A80F0137 80913720 BE49081D 1725BC31 D29EE131 71312296 2B031729

1AA00 - 66DA14F1 719667F5 F9173170 17153931 049EA3F3 1A59E6E0 86A5E172 637F31A6
1AA40 - 9E25D31A 89629C31 4B9E6008 EA3014BE 203BEAFC 7E10D791 0E74107A 900006A9
1AA80 - F17058F8 5A52F175 D014B137 C2135611 F8F29D80 D41018F0 60503F44 546402B4
1AAC0 - 549502D8 8F12450D 479F031C 27021877 D2132121 D88E1A6A 1111371F 1C6F2141

1AB00 - 1351307C 695C6133 13110017 1D214FCA D8840317 21701711 338B8311 3114B966
1AB40 - CE850302 71B0D412 01311711 4B3150B6 E1711574 AC717081 68165B01 81AC2573
1AB80 - A4E49115 372F7600 17F5CE8D 6245080D E0D4A015 3178EF7D 30ACB8EF 65A8FE1C
1ABC0 - 60877008 DF010113 6067D200 713480FF 7B00AEA8 0DF77AF2 03172860 503028D8

1AC00 - EC20100D 1AE88408 4184231B 79665085 03422DF1 C9C91341 4A3302D7 70445F13
1AC40 - 1839ED63 851B619E D0D852B6 157C3166 21861008 6250B66B F0BF0AEA 2302110A
1AC80 - F18F4FF5 031322FB 950DB950 D9293FAE 5AF40300 13610A8E CA7F1371 351098FE
1ACC0 - E010402D 4773FA96 8EE28DBF 6BF60D0D 5FE2011A 1348408C 7C4D7730 DB1B688F

1AD00 - 21448577 2AD8F71B 608FDA67 0511135D 014B7C30 5BD60E9A F0119779 5D6D711B
1AD40 - 7C851031 338F4DC8 040087A4 D1371358 BF9C11B8 B61C0386 A808C15F C8C325E4
1AD80 - 1184A760 08C5E2F8 508E6B6F 31D08E26 7D784F8F 13DB01B0 C8F21441 028E4D4C
1ADC0 - 8E22AD86 AB0310F9 66C98707 98632931 0F771EDB 112C2135 13211B7A 031371B9

1AE00 - 95F21448 EF74CD23 121D5221 A3A5F8FF 7C80AF2A 7E155717 00714517 50714517
1AE40 - 51B0C8F2 1461451A 995F1461 34135782 F1631361 0A1B3A5F 21421301 66747006
1AE80 - 1857B600 616B7260 DA103164 132D2312 1EE228F5 5F8018E1 421318FC 1DB00411
1AEC0 - B13411A1 35151759 11181CF1 5571C131 E014DB8A 15378CAC CD1468AE 008D3809

1AF00 - 07850071 34164132 208FB0F8 0BF61B23 FA113606 18414662 2077208F E3F80491
1AF40 - 2F308947 EEDB8AA8 0066B608 CECBC061 371F149F 2145D255 0A2EA6E8 0F004238
```

```
1AF80 - 3250A0E1 7F14517F 13614507 1E109F15 1717FAF4 151717F1 55717FAF B1557011
1AFC0 - B149F214 613518F1 567AF718 F156718F 1527AF81 8F152706 1A159F14 616F8222

1B000 - 3B8621B0 64400580 D0B26146 13407018 9F648907 08814022 AF1ADCA1 5A91512A
1B040 - 1D2EB155 61B34AB8 A3557093 931B05AD 495C40D0 2003A1DB 1C200233 03939E20
1B080 - 0F6BB6B6 201B960D B960D470 9660F200 107A34A3 4C20601A F0CE4538 0D0F6CE4
1B0C0 - D1151717 FA6E55FA 2E5FEB36 CE57E151 11378091 37200313 1134D962 101B995F

1B100 - 2146EE13 0CE4C480 D0F6CE44 218F1CF1 5271517A 6E5EEA2E 58EB36CE 50ED2809
1B140 - 132EA130 133EA131 15211511 20031311 34D96210 1B495F21 46E2130C E49480D0
1B180 - F6CE4421 52715171 6F17FA6E 5EEA2E58 EB36CE50 E1521151 11368091 34137809
1B1C0 - 13520038 21CE4158 0D0F6D5C D4A118F1 CF152715 7797600C D5AED280 9132EA13

1B200 - 2133EA13 3AF01521 AF215719 160003A6 E0004821 20AC8D1B 05326009 B676B8CA
1B240 - 8680D00D BF005A04 A9004563 05A1CB74 04592B44 0D887F17 5112000C 045FA0C5
1B280 - 7FC5D9C5 C5C10C5D E0494900 8A800F80 33200F92 26C05AA6 B34D05E0 A5455DE4
1B2C0 - 60DFA264 9C0462AF DA048212 6A9120B0 5AF2A86A 80978320 6A15A99A 15A15A11

1B300 - 070CA0C4 FDA1956F DE2097A0 0008FFAC E0059780 0BF0BF0E 4BF0E42E 570BD0CC
1B340 - 9088F200 2D8D08A9 0080FE20 5F0F580D 488002F2 B895FECA 431B0557 F8ADFD80
1B380 - DE0280DE D0CC038F 83DB0AF1 D882281D 81D137C9 C9135136 D7832715 111C114A
1B3C0 - 14F14914 C161CD5B E1CFDB13 40130A14 B9821117 131E0966 A017F17F 03B04A64

1B400 - 96CD0143 137C2137 17D03816 81681681 68168168 16018128 12812812 81281281
1B440 - 28120116 114A31F0 96637136 1098F88B 7077C58F FD870587 16813713 610B1635
1B480 - A11838F5 0A8014A3 00962351 61D2306D 514EED16 114A1613 16F962CC B645C213
1B4C0 - 6C913470 35310F76 9596C027 91C11B13 4D95C22E 33E2E18D 821E08E5 7BE8E140

1B500 - D8F13DB0 8E25FED6 0870F410 A31ED774 51371091 378F6253 07BD47B4 53211002
1B540 - 5FF0078D 45378537 61100007 D25C2304 F22E304D D392E828 2D740F1F 3007414E
1B580 - 171040D4 53890606 B1576D00 0C8863DD 32BDF14B 96650179 862607AE 37A9F208
1B5C0 - 4984BAF2 31BD2831 4F729470 F31027C8 039FF317 F787000C A87BE031 AF7760E6

1B600 - FD31A5DA 755046FC 7D40E7F9 75406EF9 7C337140 24F9D2CE 796385B7 23004FC7
1B640 - A2029CD8 79378597 9038639E D278425E 0318F762 274B2731 4007FF94 23878718
1B680 - 58390000 0E001D29 76D3AF21 4795E02F 2A861455 5C310296 2CB86851 147CE4A4
1B6C0 - 8AE80848 17907061 34146F2D 51BD93C1 14E161C5 561962E1 88FC0CD9 6E6E8530

1B700 - C5ED669E 60828FA2 4205C0BD ECCE01F4 2012022F F1FDEEA8 0EA0CDE9 EE5A0FEE
1B740 - BA0AF111 9D58EE29 0ED76D22 5310D77F 211B8E51 90C181DD 2CD44914 E962B016
1B780 - 1B565CEB F631AD26 76D21191 36109868 4117D7F2 21CD313D 74B268BE 32CF07CD
1B7C0 - 137A5A5A 53527799 28426EBD 786157D8 71757171 5BC316F7 F9075615 CD8457F5

1B800 - 177321C1 7342ED62 00E850FD C408D530 CD8519D6 10AD0200 056D8750 06F41171
1B840 - 8ED08013 51C317B5 8B8754ED 214D1181 3703D1E5 540D1875 9C1C3716 17FDF253
1B880 - 1FE71D13 19D6CD18 52873008 5385A850 AA2102D5 70918674 1860F010 A7AD0127
1B8C0 - 3C01CD17 51757F71 AD4FFBD2 FF4FA100 0311F9E2 4E9E6E08 407711D0 12286060

1B900 - 63A192D6 1D996E50 14F86940 E614D6E3 F841D2E6 86850147 1222305A 1A04204A
1B940 - 41228685 17B80751 1312D621 18517601 87862037 BF072B08 55778E64 3113710A
1B980 - 73100240 02031D26 E5B75A08 C62187BB FD68713E 7AFE1371 121C7159 387B9017
1B9C0 - 31593137 03846D00 9A860B03 D1147C91 7315D31C 38480387 88F318D5 078E06AE

1BA00 - 8CF4CC8D 16B90110 1371357C EFD61080 1129CECE 12903119 E6E61091 34031281
1BA40 - 37135128 03201C31 7114B514 133E2133 E6E6DEDE 0C137FA8 09FA8BB7 21371551
1BA80 - 200375AF 11913414 A8E31898 D3E3208C 0E4C311F 14D11011 92976651 1B71C576
1BAC0 - 55740F7C 3F277A9F 8E2E0D7D BED17C95 13517184 5866906A F6171715 F44314A5

1BB00 - C2414321 1D3A02D5 A0857D0E 20D0A251 435664D4 36434C34 059347F0 1184451B
1BB40 - 4C4154A9 00D4703D 660FE360 1F9014FD F00FC852 4274F274 62547004 D604D210
```

```
1BB80 - 00316F9E E026A5F7 00EF17DE D1181358 659E8450 36E4176C 559D240D 7F855EC1
1BBC0 - 107A841C 9143D813 3135100D 472404BB 8520C0C0 C0C0C0C0 C0C74100 0A203B2D

1BC00 - 202E2C2C 05407809 80920D1E 511051B8 548C1F1C 7944147C E4A0CE14 51C11334
1BC40 - BAEE50D8 4378348F 13DB07DC 31351C76 68E7CF31 7275E3DA 11B8F235 3015F917
1BC80 - 974D37B9 D6C49853 7D63501B 0496C808 535808EE 94CAF08F 13DB0873 D2130137
1BCC0 - C213531E 2D718146 11C114F9 671F30C1 4D58E81C 733F7D33 6BAE6BD4 73DC867C

1BD00 - 07E8C227 B7C85674 738637DA F817F723 31351C7D 4B2447C0 5AF215F3 AF717315
1BD40 - F3959A08 A750A4E1 C3949A18 7081CE4B 0CF8AFC0 E7E6A4F8 50872F17 A95E25C0
1BD80 - 95C70D2A 4FE3570C BAD1109D 23141EB4 908F157C 0AF4D98E B62F5A0E 4BD4B54A
1BDC0 - F8DA8620 3D295984 1297E35E A9587084 2F8AB8D2 A3E8BA90 84168B41 1995DA09

1BE00 - 4A415C09 4F6E9307 0CE4CDD7 04735232 33F94D50 BE686280 AA284214 51CFAF91
1BE40 - 55713186 1C684387 040E7101 D17CEB44 B601DA40 35A70855 70D40700 D860E222
1BE80 - 034450A5 81005A40 25E00003 16E9E2CB 85205870 B4CD5641 C757A603 C1711331
1BEC0 - 3176B1D7 5C885378 744DCCD0 5E5046C7 F1CB8636 F870708A BCEDB798 474947E2

1BF00 - 443B8557 0EC51F24 86141761 45B9862A 98428418 70902187 14E11880 FF108470
1BF40 - 86064863 76713184 305CE044 8E85375B A80DF6D8 C2186371 8712A118 80FF8816
1BF80 - E80FF636 C2E76E01 47871708 7240F213 392A702F 8506F3C7 012BF223 B9686270
1BFC0 - 303F2155 755017B1 72D61376 B3C72204 438EB32F 4D083180 8CCA2D7C 201C466D

1C000 - F7820135 1537309B 020165D0 13324131 E2674A76 D95B01C4 7140EA14 71416910
1C040 - 7F107289 86A9C177 1C414717 4133131C 20373991 741470A0 371A9133 67891188
1C080 - DC4B9075 70596766 28EBA4C1 0B133131 D88E333F 8EA0BC75 BFD97D4F 421866C2
1C0C0 - 8EF40C84 603B0496 841B0496 C11863DD 866DD863 ED8CCF3F 7B005347 54F60AA1

1C100 - 1B13414A 310F9E20 01618F50 45059E7E D81777A2 F1353132 1C114B03 962808EE
1C140 - 7CB208C0 5FB1378F DA6705DE D7809135 880B21C5 77EE1C61 37D75911 C3310E14
1C180 - B96670CC 1491C3D2 15F305CE 0415D350 01731471 C315D3DB 13503738 88E20CB6
1C1C0 - 9D97EAE1 CF157703 109AF196 CA094850 A4D8E4E7 F201BD93 C17A3F16 114E96A3

1C200 - 69662F87 61177310 457D2C64 D98717F8 707C7701 5DE749F9 6EB02894 A402A3DE
1C240 - 416E4D29 4E66615D 7772ED58 50697963 988769F3 10D96213 E6962723 16E9E242
1C280 - D2303949 40E605E1 0443179E 08561731 CB664F11 9135D20B 109228FF 2BC00413
1C2C0 - 71290B94 D50A6620 31F25C18 F3C3908E E52C8568 511196EA 88CBF1F1 338EAF6F

1C300 - 17409155 3D6790D8 C712CD4A D0A345A0 93850A5C AB4E4031 1B05CE04 5808E3CE
1C340 - 010B01D3 CFD2312D 7CDD1719 66E01751 5F31C72E 31108BB6 0DBD305C 18AB00E3
1C380 - 03111131 8E886F11 B8FD3DE0 D956A854 414E2227 235D4A55 434A2A50 525B4842
1C3C0 - 4E500001 36061B49 5F2146D5 13334811 00EA8B42 51311003 6F080100 155717F1

1C400 - BC43E2D1 73301B06 2E278203 04A851B0 61E27710 11013107 1348CB3C D8C54BB1
1C440 - 8114E14D 17113613 4965BE03 B309E043 7E79AE8F 83DB0137 D7C21351 B401E231
1C480 - 067A301A 002E7230 1A003E31 C472208F 0AB10851 8FBAB101 AA49F30E 15406CD1
1C4C0 - D51C1137 8B3B1137 14B14816 11361349 650E0313 7D047E8F DF8E017F 7B808E12

1C500 - DE400631 184227DD FD6C64FE 10076807 55084811 07271DAC 47160D68 FA90F011
1C540 - 0130CF4C 1AC215A0 8E2FBA14 91711606 4EF11913 5208ED1E E8D82860 1371F0A8
1C580 - F2145011 361BB98F 2144011F 0A8F2147 135018F1 3DB0D230 A8B656D6 80D00D4A
1C5C0 - 5133C213 1AF01531 135D1207 80096800 56F8ED8A E401BF1A 88BF4BF4 018EAAC8

1C600 - 3314648E 07AE4C03 173B6A55 D8C38CC4 7E8EA617 E72EC747 F1F178F2 D4141161
1C640 - 8F871F08 F83DB025 A802081C 1B178F21 46D58587 4301311B E95F2142 130CF4B1
1C680 - 181AC214 A765FD41 59017054 E8C170EC 0101D910 8D78A2B5 8E4F4114 67F11412
1C6C0 - 1421298B E201094A 3DFD97EA 04615A21 19D78F98 4018BF91 32EFB8FC A901137C

1C700 - A102501D B110E2D7 011378BA FE135173 1573F2F2 1758BF4E 13310313 51771371
1C740 - 35868A01 108B2921 18D71377 180119D7 4B0D97E1 05D911B1 1242A031 C38C055B
```

```
1C780 - 038BFBF1 3514B968 00D58FBF B604ED8F 91F60868 AD832A03 1D366CFD AD23102C
1C7C0 - 98BFCBD5 D2305C11 188B11AD 658A1351 4B96800D 58FB1F60 8BB9E018 DCF66041

1C800 - 1728D8F5 9B905AE7 56D8F36F 90489133 727DAF22 480FFAF7 8E219AB4 7A47813D
1C840 - B8FA90F0 80D00DAF 415111CF 601D8117 B8C40C41 1783DD41 CF8DF3FB 012910B7
1C880 - 11676664 606CC613 710A7025 16FD015A 317D8687 38F95011 43172A54 623020E0
1C8C0 - E5C11341 6E15242F 30398280 16814215 9372D47B F5113151 78FCAA21 8FD06218

1C900 - F92231AF 9108AF68 F53331AD 0AA08F40 331AFA11 88FB3231 8F6CA217 D7417817
1C940 - 815D71C3 143D2E62 39160120 33C89415 D3D2B261 CD15D37C 4416F164 152416A1
1C980 - 42D22F31 15946603 10DA445F 118F15A3 8F733214 28316060 B56055D8 AF013213
1C9C0 - 0CA101D4 EAE481C2 4908A01D A46E5614 FA6ED612 98EC180D A1215C0C CAE0A6CE

1CA00 - 41791593 20D23102 C918F18F 8FD3DE01 36108134 1C914B70 B496CE02 08F81870
1CA40 - 56073131 5D711813 47FA7759 31771593 75021C51 49842774 3171D014 7AEAF6F6
1CA80 - 9EAE0754 47844657 4B6A7513 17796C31 15B37AA3 8AE607CF 21C315D3 111D8715
1CAC0 - 786240D0 EA570852 F87537D4 1017BB17 B0375F31 5937BA21 1213020D 23144CA1

1CB00 - 3178E2D6 B94BB4A6 A1593721 4531DB57 D2556062 9379347E 3348E239 18E01D24
1CB40 - 747466DF 2F30175A 542F7635 7A9647E7 E5540E70 217B4145 D71864EC 754547C7
1CB80 - 71172314 CB119737 68724002 81E81ED7 AF072454 FDCF56F7 5857CF41 DC571945
1CBC0 - 6066F27D 9372A248 E75E471B 07811490 831E01D4 47BC36D2 F75C471A 078F049B

1CC00 - 831505FD 862F2119 70F581E8 1ED7CF4B 185177B4 4E88617B 2791C0B5 4E75747A
1CC40 - 63727117 5D215F3C 6111CA10 11C314BB 64149670 E11A1351 79D015B3 713172D1
1CC80 - 8A900B64 0111A134 D2203184 017A1117 5D215F38 62B01117 F55E2C60 1AD3D7A7
1CCC0 - 7817CF42 115A7167 75145FE0 1ACB81EA F080DF0D 4A015216 9F303821 AD3D7A77

1CD00 - 817CF412 85170D34 00861F31 5E716727 9120E008 1FA4F45C 85178A34 0086171A
1CD40 - CB80DFAF 21561279 121A210D 0016F164 AF215641 6AA46458 A4640014 61361841
1CD80 - 36A46560 F60194A0 016415E7 0211A135 7B31AF21 5D903203 4A820001 11213111
1CDC0 - 11300170 F2710320 37840534 6527017A F22033CB 202301AF 180D0F6C E4511527

1CE00 - 16FA7850 FB755AE0 D421AF01 521A7855 0B75AF42 7BF40D5A F277010A 98B94B94
1CE40 - B94F423A 10500B14 018DE6CE 020345D0 E08B2003 2FF3E201 725F4009 126062C0
1CE80 - 7A5F4009 162F7A22 6B42047F 005A01D0 46791682 01F410C2 1574A4E0 185E7E00
1CEC0 - 71107B30 8C248D8D E92208DA 8C411731 7B17F179 17F0120D 23184D53 27088DB7

1CF00 - 91120327 088D14A1 11B129F2 11115071 6F112150 7018CA6A E8DBD410 8D6A4101
1CF40 - D5464701 DA56C701 D6464601 D816DD01 DF365D01 A9500271 1A135173 136729F1
1CF80 - 34D214F8 0F58FD3D E01C114F 10A2D314 F13E80DA 68201A16 00266ABF 13374F01
1CFC0 - 31286500 27734F13 F80FF109 8E37CB7F 4F8F92FE 0313C8E6 11853120 33610028

1D000 - 8FCB3901 19223100 80DF8FCB 390791F1 B129F215 2710116F 15271020 213710B7
1D040 - 0CE11B8C 48CA1DD5 747F7920 75DE8FE2 12071DE8 EA2AE622 10B24103 6F000055
1D080 - D8D67FE0 75FF0230 C1F410C2 15501D00 157717FA E214D15F 303A826A DF30A63D
1D0C0 - F8417670 30E65CF8 417A3015 B7038417 73015B70 38417A20 15970384 1772015B

1D100 - 70384170 10159303 2031486D 20203178 6320BCA8 0DF36071 72746F00 72000120
1D140 - 31F01F01 0C214D1F 8F3E226A 92A2E15D 515F5B26 91ECE201 F010C214 7AE5F6F6
1D180 - 0E613070 E0590E72 96DD11FD F3E21570 A065BC1D 64076B0E 1CF031D7 4A06A064
1D1C0 - AE871F0A 061D844B D54D1F41 0C21572A 26A26A26 480A064B D84103AF 07CEE400

1D200 - 68EE8DC2 DE08DB1D E08DA0DE 062CE737 C20D2314 78F5A210 4E4D5D23 152C98F1
1D240 - C480767B 11A15471 6FD21441 6F146D51 3220D230 5CAE17A9 FD4100CC 5A07CA66
1D280 - 7DC34054 1074CBE4 118E2727 FDA7C6F7 16FDA101 42D74667 A3C13710 A1377C8D
1D2C0 - 5A074666 1FB787CA C210C779 B42E7ADD 11A13476 7641D77C A70FB157 71139783

1D300 - 1972E01D 947AAC6D AF74AD77 E6D8CAAF 3D7C7817 119C9134 70DE8BA6 0614FCF4
1D340 - 41769D4D 11587167 6CEF81FA 4FACB441 729D404A CB80DF15 017D3D77 86134119
```

```
1D380 - C2136756 A11A1357 94B17515 F323912A 07AAB691 07B0A171 14F7D962 F30F10C7
1D3C0 - 52215279 7C4111A1 357F0B15 37150720 8F0BF904 D17AF113 21378A2E 07E251DB

1D400 - 3673C119 1351CF1C 41478AE5 611A1347 F365A071 05643B83 1E074F41 DD36DFB1
1D440 - 59317314 DD81C31C F208FC63 A0153797 6B133C02 E23911E0 77B46AEA 620E11A1
1D480 - 3416971C 51851117 F7D13616 71368B29 D1371C71 337DE670 A98A950B 6414811C
1D4C0 - 94AC0726 A437DDAD 2E678164 717AD817 314B1C11 4D9EE916 D01743A3 38316064

1D500 - CD66B972 4B48E765 AAC210C7 5594B278 9B11A134 AF37DAB4 71AF2A6E 75F55E01
1D540 - D34717A6 77E7D8B4 7FAF2A7E AE272D54 FD2F303A C77E6B48 D76B549C A4F5EE7A
1D580 - 5B44CAF2 26A1E70A 54DA744B 49E78A57 93B4ED15 8723A9BA 12550B16 BF7BF7BF
1D5C0 - 7F7A1B55 0B16D7F6 F6CB8FD0 DE0962B1 7B696ADD 11913418 418F18F0 1681D112

1D600 - 131179D2 15F3C613 416411A1 36111131 1CF1C414 3CC2391C E417F174 D7D22030
1D640 - 5D52315B 31738E2D 3AF0F4E5 E5E5E5D9 B144A181 CC4C4C81 37C21378 B3BCDB16
1D680 - F16F1691 5A7103D8 1111311C F1532A2C 5D0D9137 17413792 C92D7135 177119CB
1D6C0 - 13413717 7DF8E83A D11B15C7 DB111131 1C4143E2 D5137CA1 371CF1CF 13710A75

1D700 - 528C183F 0008FB7E 2A86EAF2 A7E4F04E 18ED845E AF21098E 067F1191 DF594A50
1D740 - 1C079095 F085E726 98C967F8 EAE7F8E3 66F4AC91 2C08EEC7 F6CBF8E3 66F47E91
1D780 - 6AE77097 B6A47D7F 2940D111 7D4946C7 0694FB7E 8948B5F9 00000084 28EAE6F5
1D7C0 - 118C9D6F 8E6C6F85 220D231D 68E617F4 606C9A13 710A1D06 7C585606 25F8E647

1D800 - F8E966F4 4E7CA811 A1347841 43D74988 622411A1 3416C16C 156710B1 1B96A91A
1D840 - EABF6BF6 10B8FE3C 1064EF1A 5E00268E B07F11A1 3474F1AE 511A1341 F8500013
1D880 - 3CA13314 1173AE91 4D16971B 11361641 3617B145 167AF215 E78F5E03 1BF6BF61
1D8C0 - C915D916 715671C5 1CF15578 40208F64 760862D1 8F534108 F856608F FAD108C5

1D900 - C5F8EAF5 F8508DA0 860119DA 78E8CA13 17FD88C8 87D8E2D5 F1111311 C41CF1CF
1D940 - 133D81B1 75F2142C CCCD6E12 08DC3310 AF372B84 0079D177 A8400AEB A6E4D096
1D980 - A808CAB5 F7AB1788 84007FA1 2F304AC5 75784007 C91A4D51 F23A9BBF 7BF7BF7F
1D9C0 - 77E81705 84001587 23A9BA12 550B16AE 7F6BB6A6 BBF4BF4B F4F49628 08C455F0

1DA00 - 311A1341 6114AA6C 203010E0 2BE2BB2A AA3010E0 E7E61D1C 5A64540C 196C3FDC
1DA40 - 92D40EA1 63D215E3 C6017601 0E4E1504 01152416 DD015A39 4C812031 94F4F4B6
1DA80 - 2D096E40 E48F9501 18214638 ECB3F4B4 30CA880E 063030E0 13020E05 90A60762
1DAC0 - 0A892130 10313416 015E0AC9 A4E81216 E15A3030 0310102A E57860BC CAC80E46

1DB00 - 94CE1B65 4D1AC4A4 65AE1601 524B4657 DAE903AE 902AF2A7 E27A9515 E70E150E
1DB40 - 14915001 58716727 A96A1350 0B1701D0 A6E400AE A203030E 02C4C4F4 B8E80D01
1DB80 - 19CA8EA7 6FC21341 52433124 80120344 15000185 2434D55F 21341848 68008BB0
1DBC0 - 01361348 BA008C8A BE136061 3431E714 A9665016 18F725F0 07134500 8D871F00

1DC00 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
1DC40 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
1DC80 - 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
1DCC0 - 00000000 00000045 57560235 56070213 33C20213 93833302 021323A3 13030216

1DD00 - D6FF1010 B5D28000 00096027 0C110513 B1AE1602 F02594A1 2C324621 A2594EB2
1DD40 - 594A3329 37042348 54E64594 594A8401 43000006 E64F000A 70EEF900 98AEFF61
1DD80 - 078F4FD5 202AF4FD 4309505F FB30670E EF840A20 EEF15058 14FD0606 40EEF960
1DDC0 - 222AED27 0F988EFE F35221FD 3B11321F D2938221 FD161F12 1FDD908D 6CFDD70A

1DE00 - 319EDA80 0FBCEFEA 022D9F7B B0709DEF 8C000000 05D02200 FD4E0E32 1FD7F0C1
1DE40 - 19E72014 150FF6D2 0FB8EDD0 1E473FDA 84000000 C11F095F DA315CFF ED5412E9
1DE80 - EFF051E5 5CFDA61D 69BEF364 3AACEF77 153F4FD4 81850EEF 191A58DE F4A10000
1DEC0 - 00CB1256 5FF7C149 90FF4D10 00000FD1 85B9F79F 1975EFDA E15D4EFF 602E69EF

1DF00 - FF020000 00A12DE3 BEFB22AF 2BEFC327 CE9ED342 A30EEFC4 213DAFD7 52C30EEF
1DF40 - E640601F D4620000 00F62C73 0FFA7243 0EEF382F 3D8ED092 708DEF1A 2A38DEFA
```

```
1DF80 - A2000000 5B200000 0EB20000 009C2075 EFF3E238 6EFDEE20 20BED7F2 360EEF00
1DFC0 - 3F74CED1 13B47FFD 223EE2BE DB23DDF0 FDA33176 0FDD43C3 1EEF6533 C1AED563

1E000 - 685CFD87 32AD0FD5 83000000 B934D20F FB74ADD0 FD6A3473 CEDD212A 5AFF5B3D
1E040 - 5D4FD6C3 25C4FD7D 3514CE72 E3511EEF DE3B24CF D704C520 FF41434A 8FD72467
1E080 - 65FF2341 A6FFD544 003CED85 40000005 14343510 91444442 54220B14 44A41424
1E0C0 - 3530B144 4A455354 54031464 50914E47 4C454605 1435E470 B1435359 474E4805

1E100 - 1445E490 5249554A 07341445 42B09348 41494E41 1F348414 25355445 4221D348
1E140 - 41425355 44501934 C41494D4 31934C41 43535419 34C4F434 B451B34C 43545144
1E180 - 561F34F4 E4452514 35457173 4F4E4458 1734F425 2591B342 55414455 4B1D4454
1E1C0 - 641455C4 45D19449 43505420 574425F4 05E17444 58442F1D 54E444C4 94E45402

1E200 - 554E474F 09542525 D4421295 48514344 53295485 05D41342 F548505F 4E454E44
1E240 - 552B5485 4554E444 62564948 5D0764C4 14747296 4C4F4F42 582764C4 F4759276
1E280 - 4255454A 2B744494 350542C2 97444943 505B2584 4544D279 4E445F4E 2DB45495
1E2C0 - 44546442 F2DB4549 544F475E 4033C434 135C4744 5237C4F4 34B4339C 4F474051

1E300 - 3437D414 4584637D 45414E47 35D454D4 839D4542 5745493D D494E425 5414C4A3
1E340 - 5E414E4B 37E45414 25C35E45 474D3705 342544E3 9055454B 442F3905 C4943545
1E380 - A1705F4B 45404505 F4051450 5F43524D 05259465 14455434 D0525F44 55434454
1E3C0 - 45055545 54B05759 44445846 4F2514E4 44F4D494 A5745255 44484B25 54E414D4

1E400 - 5494F255 4E455D42 45425A49 25543554 45B4925F 455E444C 45353494 E0735445
1E440 - 465D4B35 54345525 54F4D355 44544144 55415D35 54454594 D4542573 584F4753
1E480 - 57351525 4545D354 51425455 50555535 4544C094 5F44514C 465F4525 14E43564
1E4C0 - F4257574 52514058 5F55E405 25F44554 3495F55E 43554345 52554A57 65142535

1E500 - B5765542 54222975 94444584 53B7594E 444F475E 47A55425 F4C11FF0 000FF000
1E540 - 00000005 0660CB05 514B13E1 E35202E3 56625825 D2713F33 E63E354B 31346A4C
1E580 - F4A25335 E35E35E3 50E09000 003BE3FC C1F856E0 00023C0F 00003EB0 F0000ADB
1E5C0 - 0F00001D B0F00008 CB0F0000 FBB0F000 06BB0F00 00DAB0F0 0004AB0F 0000B9B0

1E600 - F000029B 0F000098 B0F00006 CB0F0000 DBB0F000 04BB0F00 00BAB0F0 0002AB0F
1E640 - 000035B0 F0000A4B 0F000014 B0F00008 3B0F0000 F2B0F000 062B0F00 00D1B0F0
1E680 - 00041B0F 0000B0B0 F000020B 0F00009F A0F00000 FA0F0000 D2B0F000 042B0F00
1E6C0 - 00B1B0F0 00021B0F 00006DAD E000000B 0F00007F A0F0000E EA0F0000 5EA0F000

1E700 - 09AADE00 003DA0F0 000ACA0F 00001CA0 F00008BA 0F0000FA A0F00005 6C0F0000
1E740 - D9A0F000 049A0F00 006BA0F0 000DAA0F 00004AA0 F0000B9A 0F000029 A0F00009
1E780 - 8A0F0000 08A0F000 077A0F00 00E6A0F0 00056A0F 000013A0 F000082A 0F0000F1
1E7C0 - A0F00006 1A0F0000 D0A0F000 040A0F00 00BF90F0 000F8A0F 000068A0 F0000D7A

1E800 - 0F000047 A0F0000B 6A0F0000 26A0F000 095A0F00 0005A0F0 0074A0F 0000E3A0
1E840 - F000053A 0F0000C2 A0F00003 2A0F0000 A1A0F000 011A0F00 0080A0F0 000FF90F
1E880 - 00006F90 F0000DE9 0F00004E 90F0000B D90F0000 2D90F000 09C90F00 000C90F0
1E8C0 - 0007B90F 0000EA90 F0000809 0F0000FF 80F00006 F80F0000 DE80F000 04E80F00

1E900 - 004790F0 000B690F 00002690 F0000959 0F000005 90F00007 490F0000 E390F000
1E940 - 05390F00 00C290F0 0003290F 0F4283FF EFCD1CEE FEF0E200 ECEF614D 87DEF5C3
1E980 - 108DEF9F 0CC7DEF6 1281ECEF 47160ECE F5807DEF EF662BF2 CFFE0316 7DEFD711
1E9C0 - 15DEF881 6C4DEF4D 03024FF7 C00624FF B73D16DE F0000320 EFDD4891 4FF0000C

1EA00 - FAF00006 D0DE0000 2A1DE0EF 360FCEF0 0091CCE0 D2353CCE 0000C1CC E000053C
1EA40 - CE0000F3 CCE00009 4CCE0341 BE6DE000 0B5CCE00 00000000 000C5CCE 000094DC
1EA80 - E06202FD CE00A171 ECE0E53A FDCE0000 C970F000 03970F05 C2894DEF 1B2F84DE
1EAC0 - F864BA6D EF8B2594 DEFBA181 4DEF3C40 214FFF54 B63DEF3B 0673DEFF A4183DEF

1EB00 - F20CC2DE F900DD2D EFA308E2 DEFD3244 DFEFAF2A A7FEF134 FB7FEFD8 34E7FEFF
1EB40 - 14D7BFEF 944D8DCE F0005BDC EF63316E BEFD900C 2DEFA253 12CFF294 ED59FF00
```

```
1EB80 - 0B76DEF4 B19BCFEF 582593DE F0000A60 FF5058AE BEF50371 DFEF1F23 FCFEFD52
1EBC0 - 1EC4FF56 3ACC4FFC F4ABC4FF C41FBC4F F6421AC4 FF0008F6 DEF000FE 6DEF8A0C

1EC00 - C59EDEC2 4F7FEDF0 1D4D7E75 51219BE7 FD06DCAF D000082B F06A214E 7ED9D34C
1EC40 - FBED713B E6BEDE11 AC5CED79 2F8F9FDE 82D588ED D94111AF 8000439A F03D1DD9
1EC80 - 9E92236C B9E98314 A29FDCB0 88D9E8EC 303F8FDF B12F68E7 F120F99F DC2264FB
1ECC0 - ED254D3F BEDF2124 FBEDA937 329FD174 433CED17 20000001 70000000 28300000

1ED00 - 05D20000 008E065B 4FD4B3F5 B4FD4103 51FED201 E710FDE6 31A98EC3 35E100FD
1ED40 - 7843598E DB613F88 ED704B82 AED3E1B8 C8ED0F13 9C8EDDE3 A4C8ED20 2804AED8
1ED80 - 432B09ED F335029E D2154609 ED391BAC 9E00D42A C9E0C72E 1D9ED4E3 09C9ED65
1EDC0 - 278C9E0B 504ACFED 05057C9E 0FE457C0 FD7A3462 BEDC70E3 67EDF437 C94FD540

1EE00 - B9B7E700 0D039ED0 00BC30F0 0002C30F 00009B30 F08E4000 0008B47A 30F0061B
1EE40 - 83AE0C74 CDB9E06A 47FB9E0D 10EEB9E0 660B4F9F D0906264 FDC34ED5 4FD000AC
1EE80 - B9E0D151 CB9E0824 ED08ED90 2FAB9E85 1424352A 71434F43 5A951444 445D514C
1EEC0 - 4C48F514 E444B871 43594E49 97144514 E4B97145 545F4EE7 24143554 9E724545

1EF00 - 4058E734 14C4C49F 73414254 40D53414 45CE7345 494C4279 3464C414 74AF7348
1EF40 - 425424A7 34F40595 5B534F43 57974414 45146C94 41445544 28774414 45547754
1EF80 - 454649BD 44547425 5454353D 5445474F 694454C4 14956DB4 454C4544 5547BD44
1EFC0 - 54354525 F495EB54 494D4CC7 44943505 5C544946 56854465 A51B7544 494458B7

1F000 - 54C43554 5F554E44 4AD55405 35177542 525C4577 542525E4 67954252 5F4253E7
1F040 - 5485F425 C8554850 54976414 34458A96 45445348 48C364E4 00564F42 53C36405
1F080 - B6974F43 55524CD7 74F445F4 DD374F40 039464FD 994D4147 454FF594 E4640799
1F0C0 - 4E405554 59CD94E4 45547454 25AC594E 445C9594 E4852B39 405A6394 357E5946

1F100 - 5C4EA7B4 54954237 7B454953 5FC5B454 955E5C45 4E49A5C4 54450CBC 494E4055
1F140 - 545FB7C4 943545BB 3C4E4199 C4F47413 03395C4F 474093C4 256B7D41 494E42DD
1F180 - D4148525 5414C4C6 5D41485D A7D45414 E4D95D49 4E4CA5D4 F444477E 414D454D
1F1C0 - B7E45485 454C5E4F 445185E4 55D43A5F 464641E3 F4E40EBF 4054594F 4E4DE3F4

1F200 - 25D85F46 564FA905 14553554 7D305949 7705F425 451D9052 5544465F 99052594
1F240 - E445DC90 55525745 4BED2514 449414E4 354D5251 444E6725 5414447C 7255414C
1F280 - 4CB52554 D46ED255 43545F42 554ED525 5435F7B2 55445552 5E4BD525 D444D652
1F2C0 - 5E4440A5 2555E4EF 73544546 5E993564 C41474BF 53574E41 A93584F4 2545BC53

1F300 - 594E4695 35152529 73545144 5EC73545 54056F73 545F4059 D7354525 426A5355
1F340 - 5241C545 14247F54 514E4897 458454E4 4F94594D 45442599 4594D454 254E7459
1F380 - 4D454B73 45F43F94 52514345 4AE555E4 640B9550 5253442B A7553554 252E9553
1F3C0 - 594E474D F56514C4 5A775149 4458D1FF 54252502 C45000C8 110855E6 46562766

1F400 - C6F677C6 1207F467 562766C6 F677CC13 0A548505 F4E454E4 45820392 C1140345
1F440 - 14E4D3E3 FC015040 3E5E6567 6CC06020 3E503CC0 70203F20 3C01804F 2A55627F
1F480 - 6CC190AE 45676E5E 4F6E6D29 6E647C61 A0735152 582E6567 692CF0B0 ECE21427
1F4C0 - 76C21C05 C4F47482 0392C61D 07C4F474 82E65676 92CE0E0E 3F0F2E3F CE0F0E3F

1F500 - 0D2E3FCD 001E3F1A 203CD011 113E5E3F CD021E3F 1E503CC1 31A35967 6E616C65
1F540 - 64602F40 7C814185 5E6F6274 65627564 6C415169 4E656871 63647CC1 61AC4F67
1F580 - 70224164 74756279 7CF171BB 35973747 56D60254 2727F627 CD281A94 E6375766
1F5C0 - 66963696 56E67470 2D456D6F 62797CC1 916D4F64 657C6560 2E6F1564 6C12A1BC

1F600 - 34F6E666 96765727 164796F6 E6CD0B1E CE11464C 81C18355 72637362 7960747C
1F640 - 51D15255 636F6274 6E5FC11E 133547D6 47E8EC81 F1844164 71602459 70756C41
1F680 - 026E4F60 24416471 6CD01216 4E4E8ECB 022085E1 2C313248 575F4254 4E8ECB24
1F6C0 - 2A051627 16D65647 562702D4 69637D61 6473686C 51525354 72796E67 6E5FC716

1F700 - 26E457D6 56279636 E4FC7172 EFE3D416 E697E4F0 37C5182E FE264567 7E4F037C
1F740 - 31924348 6E6C632E 8EC81A22 64F425EB E3E45485 45C81B23 E4548545 EBE264F4
```

```
1F780 - 25CA1C22 2545E4EB E474F435 5524C31D 2ECE494D 4147454C 31E2ECE4 553594E4
1F7C0 - 74C31F24 94D41474 54E5FCF0 03ECE245 1424CF01 32355726 E8EC1123 36516270

1F800 - 2E9EC513 3EDE5021 42727169 7C3143ED E4963747 9636CF05 3EDE202F 407C7163
1F840 - 654E6460 2F66602E AECB073E CEE1FC71 83E1F602 641696C6 5646CB09 3EAEE8EC
1F880 - 41A3ECEE AE337075 636C71B3 EAE60254 87963747 37C51C3E 6E514363 6563737C
1F8C0 - B0D3EAEE 8FC31E3E AE402F40 756E6C61 F3ECEEAE 40245970 756C5104 54456679

1F900 - 63656E8E CC1144C4 96E65602 EFE3C4F6 E676C812 44752796 4756E8F1 5646CB13
1F940 - 48E4F647 02458696 3702EAEC C144A655 62796669 70264169 6C6CF154 BB55E6B6
1F980 - E6F677E6 02341627 46C81648 25F27502 542727F6 27C1174E FE364163 747C1184
1F9C0 - EFE335C6 F677CA19 497527F6 E67602E4 16D656C5 1A4EAE00 2EFE2249 676C21B4

1FA00 - 53597E64 71687CB0 C4092E7E C31D4415 57F64756 E7ECF1E4 BB548736 56373702
1FA40 - 34861627 37CB0F4E 6EE9EC11 05ECE354 870727C1 115ECE30 51627D6C F125BBD4
1FA80 - 96373796 E6760205 1627D6CF 035ECE26 51627CA1 45905275 63656465 6E63656C
1FAC0 - F055ECE2 B45697C7 1656F407 562716E6 46E7EC91 757F4075 6271647F 627E7ECF

1FB00 - 18584564 D4027525 E402C4F2 0A3F1CB0 95E6FE7F C31A5E6F 40234162 746CF0B5
1FB40 - 2752747E 0FCF0C52 656697E0 FC11D532 5561646E 0FC11E53 0527F647 E0FC11F5
1FB80 - 355E6072 7E0FCF00 62341647 E0FC9116 34527B60 2F3344F6 E656C615 E4028247
1FBC0 - 27B6E7F0 92C616E7 94C6C656 7616C602 C817E802 54870756 36475646 CA18E902

1FC00 - E4F64702 64F657E6 46C419E6 34F6E647 568747CE 0AE36496 C656C01B E40277F2
1FC40 - F602C61C E794E667 16C69646 02C11DEE CE335471 647CE0FE 345F6F60 2CB20FAA
1FC80 - 30214C69 676E6024 786566E6 0254E444 C4E4C811 F8452716 E63766F6 27D6CC03
1FCC0 - F294E666 C214F502 94E60757 47C015F4 02F46766 C6CE06F3 0557C6C6 C317F002

1FD00 - F32F6660 2F2C618F 7020527F 64756364 7CFF1575 54254595 5594F405 738393F2
1FD40 - 14354464 7484A4B4 C4D34353 63A2A585 346524E4 D48292D0 132333D2 E00000F0
1FD80 - 80900221 31D003E2 C2B2FD4F 5F3C3F4C 9B5E5D6B F9D9E929 9FCDBDDD 9CDC5CCC
1FDC0 - 8E8A6979 89498BCE DBBE8CBB 7BEE5BF7 99A9B909 81000011 70002010 40F730B0

1FE00 - 71011777 56274797 5796F607 72B7D7E5 16374666 7686A6B6 C6B34252 62A3A787
1FE40 - 366726E6 D6B5D591 12223204 610000A0 5060A141 5191C0C3 E3F31202 20322420
1FE80 - 58098072 08209203 807801F0 280E2348 00331332 33333433 53363373 3833933A
1FEC0 - 302F0C32 D32E32F3 20401412 41341441 54164174 1841941A 41B41C41 D41E41F4

1FF00 - 10511512 51351451 55165175 1851951A 51B50680 D5008011 0D044144 5140220F
1FF40 - 0E1F0E00 11124143 59434441 2E512E61 2E712E00 050B4549 502022C0 2ED02E44
1FF80 - 05045548 54502210 001D0E44 050C4946 41302110 00220503 54414451 410D0E00
1FFC0 - 1112494E 40202440 2E502E60 2E702E00 150C4548 50202480 2E902EA0 2EB02EFF

/SLOAD:  End of Saturn Loader Execution
```

```
                *     SSS   BBBB    &     DDDD   V   V  RRRR
                *    S   S  B   B  & &     D  D  V   V  R   R
                *    S      B   B  & &     D  D  V   V  R   R
                *     SSS   BBBB    &      D  D   V V   RRRR
                *        S  B   B  & & &   D  D   V V   R R
                *    S   S  B   B  &  &    D  D    V    R  R
                *     SSS   BBBB    && &  DDDD     V    R   R
                *
                        TITLE  Main Driver <831216.1559>
00000                   ABS    #0
                 PAS2+  EQU    2              Set to include pass2 deletions
                        RDSYMB TI%EQU::MS
                        RDSYMB SB%RAM::MS
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) COLDST  -  Cold starts machine
                **
                ** Category:   SYSTEM
                **
                ** Purpose:
                **      Initializes all system RAM, IO Buffers, Pointers etc.
                **
                ** Entry:
                **      None
                **
                ** Exit:
                **      Exits to MAINLP
                **
                ** Calls:      CONF,INITCL,DSPRST,WIPOUT,AUTCLR,BF2DSP,EDITWF,
                **             I/OALL,FPOLL
                **
                ** Uses.......
                **  Exclusive: Absolutely everything in the entire machine
                **             except independent RAMs
                **
                ** NOTE:
                **      This routine should be used with caution since it may
                **      annoy the user.
                **
                ** Algorithm:
                **
                **      Enables interrupt system
                **      Initialize CMOS test word
                **      Initialize system RAM to zeroes
                **      Reset display
                **      Turn display on
                **      Set display row drivers
                **      Set display contrast nibble
                **      Initialize DELAY parameters
                **      Perform ColdStart configure
                **      Create Statement Buffer
                **      Initialize clock system
                **      Check for low battery
                **      Initialize flags and traps
```

```
                **      Zero RAM between AVMEMS and RAMEND
                **      Clear AUTO mode
                **      Clear program running flag
                **      Clear don't continue flag
                **      Initialize IS-TBL table
                **      Initialize PRINT and DISP position and width
                **      Initialize ENDLINE string
                **      Put Coldstart message in display
                **      Create Workfile
                **      Create file information buffer
                **      Initialize random number seed
                **      Perform coldstart fast poll
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------   ---------------------------------
                **  07/14/82   B.S.         Updated documentation
                **
                ************************************************************
                ************************************************************
00000 20        =COLDST P=     0
00002 34EE      100     LC(5)  =CLDST1
00009 06                RSTK=C
0000B 0F                RTI                    Enable interrupts, goto CLDST1
0000D           =CKSUM1 BSS    2
```

```
                STITLE Interrupt Routine
*****************************************************************
*****************************************************************
**
** Name:    INTRPT  -  Interrupt Handler
** Name:(S) INTR50  -  Reentry point for ext. interrupt handler
**
** Category:   GENUTL
**
** Purpose:
**      INTRPT:
**        Processes interrupts whenever they happen
**      INTR50:
**        Reentry point for external interrupt handlers
**        Restores CPU registers for interrupt RAM then
**        returns from interrupt.
** Entry:
**      None
**
** Exit:
**      R4(A)=D0 at time of call.  No other registers changed.
**
** Calls:      KEYSCN
**
** Uses.......
**  Exclusive: R4(A),RAM(INTR4,INTA,INTB,INTM)
**
** Stk lvls:   0
**
** Detail:
**      Uses 56 nibbles of reserved RAM to save state of
**      machine.  Assumes that the subroutine stack has at
**      least one (out of 8) levels available to save the
**      return address.
**      This routine is not permitted to alter any hardware
**      status bits or the D register since they are not
**      saved or restored.
**        R4(A) saves C register
**        INTR4 saves R4(15-5) and D0
**        INTA  saves A register
**        INTB  saves B register
**        INTM  saves Mode,P,Carry,RSTK[N+1]
**
** Algorithm:
**      Save C(W) in R4
**      Save R4(5-15) and D0 in INTR4
**      Save A(W) in INTA
**      Save B(W) in INTB
**      Save 1 stack level, Pointer, Carry, and Mode in INTM
**      If this is not a module pulled interrupt then
**            goto INTR20
**      MP=0
**      If MP still active then
**            goto MPI
**      Set flMPI
```

```
                **    INTR20:
                **      If Interrupt Ignore Flag is set
                **            then clear it and goto RESTORE
                **      If CMOS test word is invalid
                **            then  Call WARMST and goto RESTORE if it returns
                **      If VECTOR is non-zero
                **            then jump to that address
                **      Wait 8/512ths second to debounce keyboard
                **      Call KEYSCN
                **    RESTORE:
                **      Restore Mode, Carry, Pointer and 1 Stack level
                **      Restore B(W)
                **      Restore A(W)
                **      Restore D0
                **      Restore C and R4
                **      Return from interrupt
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   ---------------------------------
                **  07/15/82   B.S.         Updated documentation
                **
                ********************************************************************
                ********************************************************************
                *       BSS    #F-*
0000F 12C       =INTRPT CR4EX                 Save C in R4
00012 136               CD0EX                 Save D0 first
00015 1B00              D0=(5) =INTR4
      4F2
0001C 1547              DAT0=C W              R4(5-15) and D0 saved
00020 1901              D0=(2) =INTA
00024 1507              DAT0=A W              Save A
00028 1902              D0=(2) =INTB
0002C AF4               A=B    W
0002F 1507              DAT0=A W              Save B
00033 AF2               C=0    W              Start misc saves
00036 07                C=RSTK                Save top of return stack
00038 80F6              CPEX   6              Save Pointer
0003C 25                P=     5
0003E 450               GOC    INTR10         Save Carry (0=Set,1=Clear)
00041 B06               C=C+1  P
00044 27        INTR10  P=     7
00046 A0E               C=C-1  P              Save Mode
00049 04                SETHEX
0004B B06               C=C+1  P
0004E 16F               D0=D0+ 16
00051 15C7              DAT0=C 8              Save misc stuff
                *  start of interrupt processing code
00055 838               ?MP=0                 Module pulled?
00058 E1                GOYES  INTR20         No.
0005A 828               MP=0                  Yes.
0005D 838               ?MP=0                 See if still active.
00060 20                GOYES  INTR19         Set carry if MPI still active.
00062 7D96      INTR19  GOSUB  RCf1MP         Let's indicate MPI occurred
```

```
191 00066 851          ST=1   1
192 00069 0B           CSTEX
193 0006B 15C0         DAT0=C 1              Set flMPI
194 0006F 460          GOC    INTR20         Go if MPI not active anymore
195 00072 6821         GOTO    MPI           Handle module-pulled interrupt
196 00076 20    INTR20 P=     0              Do transparent interrupt
197 00078 1B07         D0=(5) =DISINT        Check interrupt ignore flag
          4F2
198 0007F 15E0         C=DAT0 1
199 00083 90A          ?C=0   P              Should we ignore this interrupt
200 00086 B0           GOYES  INTR26         No, then don't ignore it
201 00088 D2           C=0    A              Zero register
202 0008A 15C0         DAT0=C 1              Yes, clear ignore flag
203 0008E 5C4          GONC   INTR50         (B.E.T.) Now ignore interrupt
204 00091 1983  INTR26 D0=(2) =CMOSTW
205 00095 15A3         A=DAT0 4              Read CMOS test word
206 00099 33F8         LC(4)  =CMOSTV        Load correct CMOS test word
          61
207 0009F 23           P=     3
208 000A1 912          ?A=C   WP             Are they the same?
209 000A4 60           GOYES  INTR27         Yes, then okay
210 000A6 695F         GOTO   COLDST         No, then machine is trashed
211             *-
212             *-
213 000AA 163   INTR27 D0=D0+ 4              D0=VECTOR
214 000AD 146          C=DAT0 A              Read VECTOR
215 000B0 8AA          ?C=0   A              Is it zero?
216 000B3 60           GOYES  INTR28         Yes, then process interrupt here
217 000B5 06           RSTK=C                No, then call interrupt handler
218 000B7 03           RTNCC                 Branch to interrupt handler
219             *-
220             *-
221             *  At this point we know that the interrupt must have been
222             *  caused by a key going down.  Since a keyboard interrupt
223             *  can occur only when all keys have been up and a key has
224             *  gone down, therefore keys that are now down are new keys
225             *  and any keys that were down at last KEYSCN are now
226             *  irrelevant.
227
228 000B9 1926  INTR28 D0=(2) =KEYSAV
229 000BD AF2          C=0    W
230 000C0 15CD         DAT0=C 14             Clear out all previous keys
231 000C4 1A20         D0=(4) =ANNAD2
          1E
232 000CA 14E          C=DAT0 B
233 000CD 0B           CSTEX
234 000CF 840          ST=0   0
235 000D2 0B           CSTEX
236 000D4 14C          DAT0=C B
237 000D7 7E50         GOSUB  debnce         Scan for new keys down
238
239             *  end of interrupt processing code
240
241 000DB       =INTR50                       (Entry needed for debugger --MB.)
242 000DB 1B03         D0=(5) =INTM
```

```
            4F2
  243 000E2 15E7        C=DAT0 8            Get misc stuff
  244 000E6 06          RSTK=C              Restore return stack
  245 000E8 27          P=     7
  246 000EA 90A         ?C=0   P            Restore Mode
  247 000ED 40          GOYES  INTR90       Hex mode if 0
  248 000EF 05          SETDEC              Dec mode otherwise
  249 000F1 25    INTR90 P=     5            Set carry if necessary
  250 000F3 A0E         C=C-1  P
  251 000F6 80F6        CPEX   6            Get pointer
  252 000FA 1902        D0=(2) =INTB
  253 000FE 1567        C=DAT0 W
  254 00102 AF5         B=C    W            Restore B
  255 00105 1901        D0=(2) =INTA
  256 00109 1527        A=DAT0 W            Restore A
  257 0010D 1900        D0=(2) =INTR4
  258 00111 1567        C=DAT0 W            Get R4(5-15) and D0 back
  259 00115 134         D0=C                Restore D0
  260 00118 12C         CR4EX               Restore C and R4(4-15)
  261 0011B 0F          RTI
```

```
262                     STITLE Entry point table
263 0011D 8D00 =bldbit GOVLNG =BLDBIT
          000
264 00124 8D00 =blddsp GOVLNG =BLDDSP
          000
265 0012B 8D00 =bldlcd GOVLNG =BLDLCD
          000
266 00132 8D00 =chedit GOVLNG =CHEDIT
          000
267 00139 8D00 =debnce GOVLNG =DEBNCE
          000
268 00140 8D00 =dspcha GOVLNG =DSPCHA
          000
269 00147 8D00 =dsprst GOVLNG =DSPRST
          000
270 0014E 8D00 =dspupd GOVLNG =DSPUPD
          000
271 00155 8D00 =keyscn GOVLNG =KEYSCN
          000
272 0015C 8D00 =makebf GOVLNG =MAKEBF
          000
273 00163 8D00 =popbuf GOVLNG =POPBUF
          000
274 0016A 8D00 =rptky  GOVLNG =RPTKY
          000
275 00171 8D00 =scrllr GOVLNG =SCRLLR
          000
276 00178 8D00 =viewd1 GOVLNG =VIEWD1
          000
277 0017F D9   =reconf C=B    A          !  Reconfigure to address in B[A]  !
278 00181 805          CONFIG            !                                  !
279 00184 8C00         GOLONG =Nxtstm    !   Next statement                 !
          00
280               *
281 0018A 8C00 =warm1x GOLONG =WARM1X
          00
282 00190 6C35 =lsleep GOTO   =LSLEEP
283 00194 8D00 =sflag? GOVLNG =SFLAG?
          000
```

```
                        STITLE Module-Pulled Interrupt Handler
                    ********************************************************************
                    ********************************************************************
                    **
                    ** Name:    MPI     -  Module Pulled Handler
                    **
                    ** Category:   CONFIG
                    **
                    ** Purpose:
                    **      Recover from having a module pulled.
                    **
                    ** Entry:
                    **
                    ** Exit:
                    **      Through COLDST if configuration believes a coldstart
                    **      must be performed.  Else through MFERR.
                    **
                    ** Algorithm:
                    **   1: Set timer to make us in 12 msec.
                    **      Sleep.
                    **      Clear MP bit.
                    **      If MP set, goto 1.
                    **      Perform powerup configure (S0=0).
                    **      If configuration requested coldstart (S0=1) then goto
                    **        COLDST.
                    **      Goto MFERR with "Module Pulled".
                    **
                    ** History:
                    **
                    **    Date      Programmer              Modification
                    **  --------    ----------   ---------------------------------
                    **  09/08/82    NM           Wrote.
                    **
                    ********************************************************************
                    ********************************************************************
0019B 80A   MPI     RESET                       Unconfigure all modules
0019E AF2           C=0     W
001A1 20            P=      0
001A3 308           LCHEX   8
001A6 1BFF          D0=(5)  (=TIMER3)+7
      1E2
001AD 15C0          DATO=C  1                   Enable timer 3
001B1 186           D0=D0-  7                   Point at timer nibbles
001B4 828   MPI10   MP=0
001B7 838           ?MP=0                       MP bit still active?
001BA 81            GOYES   MPI20               No.
001BC 306           LCHEX   6                   Yes.  6-512ths = 12 msec.
001BF 8F00          GOSBVL  =WRTIMR             Wake us up in 12 msec.
      000
001C6 301           LCHEX   1
001C9 801           OUT=C
001CC 807           SHUTDN                      Sleep
001CF 54E           GONC    MPI10               B.E.T.
001D2 7F7F  MPI20   GOSUB   =keyscn             Look for keys
001D6 8F00          GOSBVL  =PWCONF             Configure machine
```

```
            000
    337 001DD 3400          LC(5)  =MFERR
            000
    338 001E4 06            RSTK=C
    339 001E6 3100          LC(2)  =eMPI            Module pulled message
    340 001EA 26            P=     6                For MFERR
    341 001EC 0F            RTI
```

```
342                     STITLE Cold Start
343 001EE       =CLDST1
344 001EE 1F83          D1=(5) =CMOSTW          Address of CMOS test word
          4F2
345 001F5 33F8          LC(4)  =CMOSTV          Value for CMOS test word
          61
346 001FB 15D3          DAT1=C 4                Initialize CMOSTW test word
347 001FF 828           MP=0                    Clear module-pulled bit
348 00202 AF0           A=0    W
349 00205       =VECTR!                         (Entry needed for debugger  --MB)
350 00205 1DC3          D1=(2) =VECTOR          Start of RAM to zero
351 00209 D2            C=0    A
352 0020B 22            P=     2
353 0020D 308           LCHEX  8                C=00800
354 00210 8F00          GOSBVL =WIPOUT          Clear 8 kbits of RAM
          000
355 00217 7C2F          GOSUB  dsprst           Reset display
356 0021B       CLDST2
357 0021B 04            SETHEX
358 0021D 7444          GOSUB  LCDINI           Set row drivers,
359                                             turn on display,
360                                             set contrast
361 00221 1A64          D0=(4) =SCROLT
          9F
362 00227 3340          LC(4)  16~4             SCROLT=4/32 sec; DELAYT=16/32 sec
          01
363 0022D 15C3          DAT0=C 4
364 00231 1967          D0=(2) =MAXCMD
365 00235 1540          DAT0=C P                Initialize command stack count=4
366             *
367             * Configure system
368 00239 850           ST=1   0                Request coldstart configure
369 0023C 8F00          GOSBVL =CONF            Configure
          000
370             *
371             * Initialize clock system
372 00243 8F00          GOSBVL =INITCL          Init clock system
          000
373
374             *
375             *  Check for low battery
376 0024A 7804          GOSUB  acbat?
377             *
378             *  Initialize flags and traps
379 0024E 8F00          GOSBVL =CSTRST          Reset traps, SFLAGs[1-32], UFLAGs
          000
380 00255 8F00          GOSBVL =C=RAME
          000
381 0025C 1D49          D1=(2) =AVMEMS
382 00260 143           A=DAT1 A
383 00263 131           D1=A
384 00266 E2            C=C-A  A
385 00268 8F00          GOSBVL =WIPOUT
          000
386 0026F 8E00          GOSUBL =AUTCLR          Clear Auto # mode
```

```
                00
    387 00275 84D         ST=0   PgmRun          Clear program running bit
    388 00278 84E         ST=0   NoCont          Clear auto halt bit
    389               *
    390               * Initialize IS table
    391               *
    392 0027B 1FD8        D1=(5) =IS-TBL         Initialize IS table
                7F2
    393 00282 AF2         C=0    W
    394 00285 A7E         C=C-1  W
    395 00288 15DD        DAT1=C 14
    396 0028C 17D         D1=D1+ 14
    397 0028F 15DD        DAT1=C 14              28 nibs of F
    398 00293 1EF4        D1=(4) =DWIDTH        Initialize DISP/PRINT
                9F
    399 00299 3606        LCHEX  0A0D460          CR LF 1 PW
                4D0A
                0
    400 002A2 14D         DAT1=C B               Init DPOS,DWIDTH
    401 002A5 178         D1=D1+ (PWIDTH)-(DWIDTH)
    402 002A8 15D8        DAT1=C 9               Init PWIDTH,EOLLEN,EOLSTR
    403 002AC 1FF8        D1=(5) =COLDSM         Point to cold start message
                600
    404 002B3 8E00        GOSUBL =BF2DSP
                00
    405 002B9 85B         ST=1   11              Don't want CATalog
    406                                            (ok to collapse stacks)
    407 002BC 8F00        GOSBVL =EDITWF         Create workfile
                000
    408               *
    409               * Create File Information Buffer
    410               *   Write File#=00 @ end of buffer
    411               *
    412 002C3 20          P=     0
    413 002C5 3230        LC(3)  =bFIB           File Information Buffer ID
                8
    414 002CA D1          B=0    A
    415 002CC E5          B=B+1  A               Length = 2
    416 002CE E5          B=B+1  A
    417 002D0 8F00        GOSBVL =I/OALL
                000
    418 002D7 D2          C=0    A               File# = 00
    419 002D9 14D         DAT1=C B               Write End of FIB chain
    420               *
    421               * Initialize Random Number Seed
    422               *
    423 002DC 1AEF        D0=(4) =RNSEED
                6F
    424 002E2 3E33        LCHEX  999500333083533
                5380
                3330
                0599
                9
    425 002F3 15CE        DAT0=C 15
    426               *
```

```
427             * Poll All Modules for Cold Start and turn machine off
428             *
429 002F7 7721          GOSUB  fpoll          Perform cold start poll
430 002FB FF            CON(2) =pCLDST
431             * Fall into main loop
432             ************************************************************
433             ************************************************************
434             **
435             ** Name:(S) pCLDST  -  Coldstart poll
436             **
437             ** Category:   POLL
438             **
439             ** Type:       FPOLL
440             **
441             ** Purpose:
442             **      Allows module to gain control at Coldstart
443             **
444             ** Should poll be "Handled" (return with XM=0)?:  No
445             **
446             ** Entry conditions for handler (registers, ST, RAM, etc.):
447             **      B[A] = Poll number.
448             **      HEX mode.
449             **      P=0.
450             **
451             ** Normal exit conditions from handler if not handled (ST, RAM,
452             ** registers, etc.):
453             **      HEX mode.
454             **      XM=1.
455             **
456             ** Available subroutine levels:  5
457             **
458             ** What registers/RAM may be used if not handled?:
459             **      Nothing matters except D(A)
460             **
461             ** Envisioned application(s):
462             **      Operating system take overs.
463             **      Initialization of buffers, RAM, etc.
464             **
465             ** History:
466             **
467             **    Date     Programmer             Modification
468             **  --------   ----------   ----------------------------------
469             **  07/15/82   B.S.         Added documentation
470             **  10/17/83   B.S.         Updated documentation
471             **
472             ************************************************************
473             ************************************************************
474
```

```
475                 STITLE Main System Loop
476               ********************************************************************
477               ********************************************************************
478               **
479               ** Name:(S) MAINLP  -  Main Loop
480               ** Name:(S) MAIN05  -  Main Loop
481               ** Name:(S) MAIN30  -  Main Loop
482               **
483               ** Category:   SYSTEM
484               **
485               ** Purpose:
486               **      These entry points implement the normal idle state
487               **      where the cursor is blinking in the display.
488               **
489               ** Entry:
490               **      MAINLP:  Almost nothing matters.  The system will
491               **               check a few flags and clear a few.  Then...
492               **      MAIN05:  Allows user to scroll displayed line if there
493               **               is one then prompts for input.  Then...
494               **      MAIN30:  Calls character editor to input a line until
495               **               special key is hit then jumps to a routine
496               **               to handle that key.
497               **
498               ** Exit:
499               **      P       =  0
500               **      Control is passed to one of LINEP,WAKEUP,ATTNTN,RUNK,
501               **      CONTK,SST,CALC,PWROFF,CURTOP,CURBOT,CURSUj,CURSDj,
502               **      CMDSTK,PWROFF,IEXKEY
503               **
504               ** Calls:       SFLAG?,SFLAGS,SFLAGC,FPOLL,AUTOCK,SCRLLR,BF2DPP
505               **              COLLAP,CLCOLL,STMBCL,NOPRGM,I/ODAL,ATNCLR,
506               **              CURSFR,TBLJMC
507               **
508               ** Algorithm:
509               **      MAINLP:If flTNOF or flMKOF set then
510               **                Go to PWROFF
511               **             If CALC mode set then
512               **                Go to CLCERR
513               **             Fast Poll (pMNLP)                          (FPOLL )
514               **             If in AUTO mode then
515               **                Go to =AUTXQ7
516               **      MAIN05:If CALC mode (flCALC) is set then
517               **                Go to =CLCERR
518               **             Clear program annunciator & status bit     (NOPRGM)
519               **             Set flDORM
520               **             If Don't Prompt flag (flNOPR) is set then
521               **                Go to MAIN30
522               **             If scrolling needed (NEEDSC) then
523               **                Allow user to scroll                    (SCRLLR)
524               **             Send prompt string consisting of           (BF2DPP)
525               **                Cursor off, prompt character(">"),
526               **                Cursor on
527               **      MAIN30:If Attn key has been pressed jump to
528               **                clean up as necessary.                  (ATTNTN)
529               **             Clear Don't Continue flag (NoCont)
```

```
530             **              Collapse math stack                     (COLLAP)
531             **              Collapse AVMEMS,OUTBS,SYSEN to CLCSTK   (CLCOLL)
532             **              Clear Don't Prompt flag (flNOPR)
533             **              Collapse statement buffer               (STMBCL)
534             **              Delete Immediate Execute Key buffer (bIEXKY)
535             **              Set "Dormant" flag (flDORM)             (SFLAGS)
536             **              Call Character Editor                   (CHEDIT)
537             **              If Immediate Execute Key then
538             **                Go to IEXKEY
539             **              If not cursor up/down then
540             **                Clear command stack flag (flCMDS)     (SFLAGC)
541             **              Clear "Dormant" flag (flDORM)           (SFLAGC)
542             **              Clear Attention Flag so HPIL won't abort(ATNCLR)
543             **              Move cursor to far right of display     (CURSFR)
544             **              Go to appropriate place to process key  (TBLJMC)
545             **                  Endline         (LINEP)
546             **                  Attention       (ATTNTN)
547             **                  RUN key         (RUNK)
548             **                  CONT key        (CONTK)
549             **                  SST key         (SST)
550             **                  Cursor Up       (CURSUj)
551             **                  Cursor Down     (CURSDj)
552             **                  Cursor Top      (CURSTj)
553             **                  Cursor Bottom   (CURSBj)
554             **                  G-Attention     (ATTNTN)
555             **                  CALC Mode key   (CALC)
556             **                  Off key         (PWROFF)
557             **                  Command Stack   (CMDSTK)
558             **
559             ** History:
560             **
561             **    Date      Programmer              Modification
562             **  --------    ----------   -----------------------------------
563             **  01/05/83    B.S.         Added documentation
564             **
565             ****************************************************************
566             ****************************************************************
567             *Note that cold start falls into this code
568 002FD 20  =MAINLP P=     0
569 002FF 7CA3         GOSUB  TNOFF?
570 00303 560          GONC   MAIN01
571 00306 6F12         GOTO   PWROFF
572
573 0030A 8E00  MAIN01 GOSUBL =fCALC?
          00
574 00310 403          GOC    clcerr
575 00313 7B01         GOSUB  fpoll
576 00317 AF           CON(2) =pMNLP
577
578 00319 8E00         GOSUBL =AUTOCK        Auto # mode?
          00
579 0031F 481          GOC    MAIN05         No
580 00322 8C00         GOLONG =AUTXQ7        Yes, supply line number
          00
581             ****************************************************************
```

```
582             ****************************************************************
583             **
584             ** Name:(S) pMNLP   -  Poll on entry to main loop
585             **
586             ** Category:   POLL
587             **
588             ** Type:       FPOLL
589             **
590             ** Purpose:
591             **      Poll on entry to main loop.
592             **
593             ** Should poll be "Handled" (return with XM=0)?:
594             **      NO!!  NEVER!!  Take over, yes.  Handle, no.
595             **
596             ** Meaning of "Handling" Poll (what does code do if handled?):
597             **      N/A
598             **
599             ** Entry conditions for handler (registers, ST, RAM, etc.):
600             **      Carry set.
601             **      B[A] = Poll number = pMNLP.
602             **      HEX mode.
603             **      P=0.
604             **
605             ** Normal exit conditions from handler if handled (ST, RAM,
606             ** registers, etc.):
607             **      N/A
608             **
609             ** Normal exit conditions from handler if not handled (ST, RAM,
610             ** registers, etc.):
611             **      HEX mode.
612             **      XM=1.
613             **
614             ** Available subroutine levels:
615             **      5
616             **
617             ** NOTE:
618             **      Machine is entering an idle state.  This is a good time
619             **      to take over.  This poll is one of the very first
620             **      things done on entry to main loop.  We have not done
621             **      display scrolling, auto line#, collapsing stmt buffer,
622             **      checking for CALC mode, etc.
623             **
624             ** What registers/RAM may be used if handled?:
625             **      N/A
626             **
627             ** What registers/RAM may be used if not handled?:
628             **      All CPU registers except D[A].
629             **      All scratch RAM.
630             **
631             ** Special memory/pointer considerations (are pointers funny?):
632             **      May be in CALC mode.  The routine fCALC? will RTNSC
633             **      if we are in CALC mode without using D[A].
634             **
635             ** Envisioned application(s):
636             **      Taking over, maybe?
```

```
637                **
638                ** History:
639                **
640                **    Date     Programmer          Modification
641                ** --------   ----------  ----------------------------------
642                **  03/23/83   NM          Added documentation
643                **
644                ********************************************************************
645                ********************************************************************
646                *-
647                *-
648 00328 D0A0 =CRLFPR NIBHEX D0A0          CR/LF
649 0032C B1C3 =PROMPT NIBHEX B1C3          Turn off cursor(Non-readable)
650 00330 E3           NIBASC \>\           Prompt character
651 00332 B1E3         NIBHEX B1E3          Turn on cursor(Readable)
652 00336 FF           NIBHEX FF            End of string
653 00338 8E00 =MAIN05 GOSUBL =fCALC?       Should we be in CALC mode?
          00
654 0033E 590          GONC   MAIN06        No, then okay
655 00341 8D00  clcerr GOVLNG =CLCERR       Yes, then resume CALC mode
          000
656
657 00348 8E00  MAIN06 GOSUBL =NOPRGM       Clear program annun. & status bit
          00
658 0034E 315D         LC(2)  =flDORM
659 00352 7D71         GOSUB  sflags        Set dormant flag
660 00356 316E         LC(2)  =flNOPR
661 0035A 763E         GOSUB  sflag?        Is "Don't prompt" flag set
662 0035E 4F1          GOC    MAIN10        Yes, then skip prompting
663 00361 7571         GOSUB  ATNTST        Check if ATTN key pressed
664 00365 8E00         GOSUBL =C=SCRL
          00
665 00368 90A          ?C=0   P             Anything there to be scrolled?
666 0036E A0           GOYES  MAIN08        No,skip scrolling & put up prompt
667 00370 7DFD         GOSUB  scrllr        Scroll left and right
668 00374 7261         GOSUB  ATNTST        Check if ATTN key pressed
669 00378 8E00  MAIN08 GOSUBL =BF2DPP       Send prompt string
          00
670                *
671 0037E      =MAIN10
672 0037E      =MAIN30
673 0037E      =WAKEUP
674 0037E 7851         GOSUB  ATNTST        Is ATNFLG set?
675 00382 84E          ST=0   =NoCont
676 00385 8F00         GOSBVL =COLLAP
          000
677 0038C 8E00         GOSUBL =CLCOLL       Collapse AVMEMS,OUTBS,SYSEN
          00
678                                           to CLCSTK
679 00392 316E         LC(2)  =flNOPR
680 00396 7231         GOSUB  sflagc        Clear "Don't prompt" flag
681 0039A 8F00         GOSBVL =STMBCL       Collapse Statement Buffer
          000
682 003A1 3220         LC(3)  =bIEXKY
          8
```

```
683 003A6 8F00          GOSBVL =I/ODAL      Delete "Immediate Execute Key"
          000
684                                            buffer
685 003AD 315D          LC(2)  =flDORM
686 003B1 7E11          GOSUB  sflags       Set "Dormant" flag
687 003B5 797D          GOSUB  chedit       Call character editor
688 003B9 5E6           GONC   IEXKEY       Process colon key def
689 003BC D2            C=0    A
690 003BE 30D           LC(1)  13           Number of first relevant keycode
691 003C1 EA            A=A-C  A            Map terminating keycodes to
692                                             start at 0
693 003C3 101           R1=A                Save terminating keycode minus 12
694 003C6 CE            C=C-1  A            C=12
695 003C8 962           ?A=C   B            Is it a g EndLine key?
696 003CB 41            GOYES  MAIN40       Yes, then okay
697 003CD 3250          LCHEX  805          Is it a cursor up or down key?
          8
698 003D2 8E00          GOSUBL =Range
          00
699 003D8 560           GONC   MAIN40       Yes, then okay
700 003DB 79E0          GOSUB  CMDOFF       No, then exit command stack mode
701 003DF 315D  MAIN40  LC(2)  =flDORM
702 003E3 75E0          GOSUB  sflagc       Clear "Dormant" flag
703 003E7 7521          GOSUB  RTNCLR       Don't abort HPIL send
704 003EB 8F00          GOSBVL =CURSFR      So video doesn't overwrite chars
          000
705 003F2 119           C=R1                Get terminating keycode minus 12
706 003F5 8E00          GOSUBL =TBLJMC      Jump into oblivion
          00
707 003FB EA0           REL(3) EOLk         13 EndLine
708 003FE 5E0           REL(3) ATTNTN       14 Attention
709 00401 EA0           REL(3) RUNKj        15 RUN key
710 00404 1B0           REL(3) CONTKj       16 CONTinue key
711 00407 4B0           REL(3) SSTj         17 SST key
712 0040A D70           REL(3) cursuj       18 Cursor up
713 0040D 080           REL(3) cursdj       19 Cursor down
714 00410 380           REL(3) curstj       20 Cursor to top
715 00413 680           REL(3) cursbj       21 Cursor to bottom
716 00416 DC0           REL(3) ATTNTN       22 g ON
717 00419 8A0           REL(3) CALCj        23 CALC mode key
718 0041C A01           REL(3) PWROFF       24 OFF key
719 0041F 080           REL(3) CMDSTk       25 Command Stack (g EOL)
720             *-
721             *-
722 00422 8C00   fpoll  GOLONG =FPoll
          00
723             *-
724             *-
```

```
                        STITLE IEXKEY
                *****************************************************************
                *****************************************************************
                **
                ** Name:    IEXKEY  -  Immediate Execute Keys Processing
                **
                ** Category:   SYSTEM
                **
                ** Purpose:
                **      Process an Immediate Execute Key
                **
                ** Entry:
                **      P      =  0
                **      D1 Points at definition
                **      R3=Definition length+1
                **
                ** Exit:
                **      P      =  0
                **      Exits thru LINEP+
                **
                ** Calls:      I/OALL,MOVEU3
                **
                ** Uses.......
                **  Inclusive:
                **
                ** Stk lvls:   1
                **
                ** Algorithm:
                **      Clear "Dormant" flag
                **      Clear command stack flag (flCMDS)
                **      Calculate required buffer size
                **      Allocate buffer                                  (I/OALL)
                **      If not enough memory then
                **        Go to =MEMERX
                **      Copy definition into buffer                      (MOVEU3)
                **      Append CR to definition in buffer
                **      Set up pointer to buffer
                **      Go to =LINEP+
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **   --------   ----------   ----------------------------------
                **  01/05/83   B.S.         Added documentation
                **  02/18/83   B.S.         Now clear dormant flag
                **
                *****************************************************************
                *****************************************************************
00428 315D  IEXKEY LC(2)  =flDORM
0042C 7C90         GOSUB  sflagc          Clear "Dormant" flag
00430 7490         GOSUB  CMDOFF
00434 137          CD1EX
00437 108          R0=C                   Save definition pointer
0043A 113          A=R3                   Recall definition length+1
0043D D8           B=A    A
```

```
780 0043F E5          B=B+1  A          Buffer size=Def len+2
781 00441 3220        LC(3)  =bIEXKY
          8
782 00446 8F00        GOSBVL =I/OALL    Allocate buffer
          000
783                                     D1 points to start of destination
784 0044D 503         GONC   IEXKYE     Enough memory? No, then error
785 00450 110         A=R0              Recall definition pointer
786 00453 132         ADOEX             D0=Start of source
787                                     A(A)=Start of buffer
788 00456 100         R0=A              Save start of buffer
789 00459 11B         C=R3              Recall definition length+1
790 0045C CE          C=C-1  A          Definition length
791 0045E 8F00        GOSBVL =MOVEU3    Move definition into buffer
          000
792 00465 31D0        LCHEX  0D         Carriage return
793 00469 14D         DAT1=C B          Append CR to buffer
794 0046C 110         A=R0              Recall start of buffer pointer
795 0046F 131         D1=A
796 00472 175         D1=D1+ 6          Point at start of text
797                                       (Skip buffer #, len)
798 00475 137         CD1EX             Put pointer in C(A)
799 00478 8C00  LINEPj GOLONG =LINEP+   Parse line from buffer
          00
800           *-
801           *-
802 0047E      IEXKYE
803 0047E 22          P=     2
804 00480 8D00        GOVLNG =MEMERX    Memory error (execution)
          000
805           *-
806           *-
807 00487 8C00  cursuj GOLONG =CURSUj
          00
808 0048D 8C00  cursdj GOLONG =CURSDj
          00
809 00493 8C00  curstj GOLONG =CURSTj
          00
810 00499 8C00  cursbj GOLONG =CURSBj
          00
811           *-
812           *-
813 0049F 8E00  CMDSTk GOSUBL =CMDSTK
          00
814 004A5 68DE        GOTO   MAIN10
815           *-
816           *-
817 004A9      EOLk
818 004A9 8C00        GOLONG =LINEP
          00
819           *-
820           *-
821 004AF 8C00  RUNKj  GOLONG =RUNK
          00
822           *-
```

```
823                *-
824 004B5 8C00      CONTKj GOLONG =CONTK
          00
825                *-
826                *-
827 004BB 8C00      SSTj   GOLONG =SST
          00
828                *-
829                *-
830 004C1 8D00 =CALCj  GOVLNG =CALC
          000
831                *-
832                *-
833 004C8 311D      CMDOFF LC(2)  =flCMDS
834 004CC 8D00 =sflagc GOVLNG =SFLAGC
          000
835 004D3 8D00 =sflags GOVLNG =SFLAGS
          000
```

```
                        STITLE ATTNTN
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:    ATTNTN  -  Attention Key Processing
                    **
                    ** Category:   SYSTEM
                    **
                    ** Purpose:
                    **      Sends CR/LF to display, Clears Auto, ATNFLG, Goes to
                    **      MAINLP
                    **
                    ** Entry:
                    **
                    ** Exit:
                    **      P       =  0
                    **
                    ** Calls:      S-CRLF,AUTCLR,ATNCLR
                    **
                    ** History:
                    **
                    **    Date      Programmer              Modification
                    **  --------    ----------   ---------------------------------
                    **  07/14/82    B.S.         Added documentation
                    **  08/27/82    B.S.         Now clears flCMDS
                    **  04/23/83    B.S.         No longer clears flCMDS
                    **  05/20/83    B.S.         Now flushes key buffer
                    **
                    ****************************************************************
                    ****************************************************************
004DA 8E00  ATNTST GOSUBL =CK"ON"
      00
004E0 400          RTNC
004E3 8E00 =ATTNTN GOSUBL =nokeys
      00
004E9 8E00         GOSUBL =GETSTA
      00
004EF 870          ?ST=1   =Clear
004F2 90           GOYES   ATTN10
004F4 8F00         GOSBVL =FINLIN        Send CR/LF to display
      000
004FB 8F00  ATTN10 GOSBVL =NOSCRL
      000
00502 8E00         GOSUBL =AUTCLR        Turn off auto#
      00
00508 7400         GOSUB   ATNCLR
0050C 60FD         GOTO    MAINLP
                   *-
                   *-
```

```
                        STITLE ATNCLR
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) ATNCLR  -  Clear Attention Flags
                **
                ** Category:   GENUTL
                **
                ** Purpose:
                **      Clears ATNFLG to inhibit effects of ATTN
                **      key.  Also returns old state of ATTN flag.
                **
                ** Entry:
                **
                ** Exit:
                **      Carry clear iff ATNFLG was set.
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: A[A],D1
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ----------------------------------
                **  11/10/82    NM           Added documentation
                **  07/25/83    B.S.         No longer clears Except status bit
                **
                ********************************************************************
                ********************************************************************
00510           =ATNCLR
00510 1F24      ATNCL1 D1=(5) =ATNFLG
      4F2
00517 D0               A=0    A
00519 1532             A=DAT1 XS
0051D 1590             DAT1=A 1
00521 A2C       ATNCL2 A=A-1  XS            Set carry if was clear.
00524 01               RTN
```

```
                        STITLE PWROFF
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) PWROFF  -  Power Off
                **
                ** Category:   SYSTEM
                **
                ** Purpose:
                **      Sends machine into deep sleep and waits for wakeup
                **
                ** Entry:
                **
                ** Exit:
                **      Exits to LINEP+ if a command buffer needs processing
                **      otherwise exits to MAINLP
                **
                ** Calls:      DPS010(DSLEEP),SFLAGS,I/OFND
                **
                ** Algorithm:
                **      Set flPWDN
                **      Call DPS010 to go to deep sleep
                **      If there is an external command buffer
                **          then jump to LINEP+ to process it
                **      If there is an STARTUP buffer
                **          then jump to LINEP+ to process it
                **      Jump to MAINLP
                **
                ** History:
                **
                **    Date     Programmer            Modification
                **   --------   ----------   ----------------------------------
                **  07/15/82   B.S.         Updated documentation
                **
                ********************************************************************
                ********************************************************************
00526 04        =PWROFF SETHEX
00528 20                P=      0
0052A 31FC              LC(2)   =flPWDN
0052E 71AF              GOSUB   sflags          Set POWERDOWN flag to indicate
00532 7340              GOSUB   DPS010          DSLEEP called from PWROFF.
00536 3290              LC(3)   =bECOMD
              8
0053B 8F00              GOSBVL  =I/OFND         Is there an Ext Cmd buffer?
              000
00542 590               GONC    PWR100          No
00545 137       PWR050  CD1EX
00548 6F2F              GOTO    LINEPj          Process cmd buffer
0054C 3280      PWR100  LC(3)   =bSTART
              8
00551 8F00              GOSBVL  =I/OFND         Is there a STARTUP buffer?
              000
00558 4CE               GOC     PWR050          Yes, process it.
0055B 61AD              GOTO    MAINLP
                *-
```

```
970             *-
971 0055F       =PWROFS
972 0055F B1E3          NIBHEX B1E3             Cursor on
973 00563 D0A0          NIBHEX D0A0             CR/LF
974 00567 B1C3          NIBHEX B1C3             Cursor off
975 0056B FF            NIBHEX FF
```

```
                    STITLE DSLEEP
                *-
                *-
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) DSLEEP  -  Deep sleep
                **
                ** Category:   GENUTL
                **
                ** Purpose:
                **      Put TITAN into a power-off state.
                **
                ** Entry:
                **      None.
                **
                ** Exit:
                **      P=0.
                **      Carry clear.
                **
                ** Calls:      ALMSRV, ATNCL1, BF2DSP, FIBOFF, I/ODAL, LOCKD?,
                **             OUT=1, PWCONF, RCLSTA, FPOLL, NOKEYS, SFLAG?,
                **             SFLAGC, SFLAGS, ACBAT?
                **
                ** Uses.......
                **             All CPU registers.  SCRTCH in RAM.
                **
                ** Stk lvls:   5
                **
                ** NOTE:
                **      This is how you put the machine to sleep.  If memory
                **      configuration changes while the machine is asleep,
                **      the soft-configured module which called DSLEEP may have
                **      moved.  Thus when DSLEEP tries to return, the machine
                **      will go out to lunch.  It is RECOMMENDED that you call
                **      DSLEEP through the MGOSUB utility:
                **        GOSBVL =MGOSUB
                **        CON(5) =DSLEEP
                **      Then if configuration changes, the GOSUB stack will be
                **      collapsed and the attempt to return from DSLEEP will
                **      give a SYSTEM ERROR.  This beats going out to lunch.
                **
                **      Secondary local entry point DPS010 is used by PWROFF.
                **
                ** Detail:
                **      Performs power-down poll on entry and one or two
                **      power-up polls on wakeup.  Control is returned to the
                **      calling routine in the following circumstances:
                **
                **        If ATTN key was not hit:
                **          An on-timer alarm is pending with program running
                **                                or
                **          A poll handler cleared =flTNOF on =pDSWNK poll.
                **
                **        If ATTN key was hit:
```

```
1031            **          A poll handler cleared =flTNOF on =pDSWKY poll.
1032            **                              or
1033            **          Password is null
1034            **                              or
1035            **          User supplies correct password.
1036            **
1037            **      LOCK is implemented with the aid of the =flTNOF and
1038            **      =flMKOF flags.  Proper manipulation thereof will keep
1039            **      the user from breaking into a locked machine.  Guide-
1040            **      lines for their use are found in the poll interface
1041            **      descriptions below.
1042            **
1043            **      Some special things happen for the benefit of the
1044            **      PWROFF routine, since PWROFF returns control to the
1045            **      main loop upon wakeup.  See PWROFF documentation for
1046            **      more detail, including explanation of =bECOMD.
1047            **
1048            ** Algorithm:
1049            **  DSLEEP: Clear =flPWDN flag (indicate that we were not
1050            **            called from PWROFF).
1051            **  DPS010: (Entry point for PWROFF).
1052            **          If ON key down
1053            **             Set ATTN flag and goto DSP040
1054            **          If display-clear flag clear then goto DPS030
1055            **             Send <cursor on>/CR/LF.
1056            **  DPS030  Send <cursor off>
1057            **  DPS035: Perform power-down poll.
1058            **          Set TURNOFF (flTNOF) flag.
1059            **          Clear MAKEOFF (flMKOF) flag.
1060            **          Turn off display.
1061            **          Clear f-g shift status bits.
1062            **          Clear ATNFLG and ATNDIS.
1063            **          Turn off timer _#3 (Low battery check).
1064            **          Activate KB row with ATTN key.
1065            **          SHUTDN.
1066            **
1067            **  DPS040: Configure.
1068            **          Deallocate external command buffer (to give poll
1069            **            handlers a chance to create one if we were
1070            **            called by PWROFF).
1071            **          Check clock system
1072            **          If ATTN key woke us up, goto DPS200.
1073            **          If program running and ON TIMER pending
1074            **             Clear =flTNOF; goto DPS200.
1075            **          Perform pDSWNK poll (who woke us up?!?).
1076            **          If turnoff flag set and ATNFLG clear then
1077            **             goto DSP035
1078            **  DPS200: Flush key buffer.
1079            **          Clear flALRM flag.
1080            **          =pDSWKY poll
1081            **          Password processing (does not require password if
1082            **            password=null or =flTNOF is clear).
1083            **          If failed to unlock machine (password required but
1084            **            not correctly given), goto DPS035.
1085            **          AC/BAT check
```

```
1086                 **          RETURN
1087                 **
1088                 ** History:
1089                 **
1090                 **    Date      Programmer          Modification
1091                 **  --------    ----------  ----------------------------------
1092                 **  07/15/82    NM          Added name to documentation
1093                 **  09/07/82    NM          Added calls to AC/BAT at end
1094                 **  09/09/82    NM          Moved puroff poll after DSP020
1095                 **  09/13/82    NM          Made CR/LF conditional on clear flg
1096                 **  09/20/82    NM          Check ON key at DPS010
1097                 **  09/23/82    NM          Clear flALRM before pDSWKY poll
1098                 **  10/25/83    B.S.        Updated documentation
1099                 **
1100                 ********************************************************************
1101                 ********************************************************************
1102 0056D 04      =DSLEEP SETHEX
1103 0056F 20              P=      0
1104 00571 31FC            LC(2)   =flPWDN         Will clear POWERDOWN flag to
1105 00575 735F            GOSUB   sflagc          indicate not called from PWROFF.
1106 00579          DPS010                         Entry past clear of PWDN flag
1107 00579 20              P=      0
1108 0057B 8F00    DPS023  GOSBVL  =FIBOFF         Reclaim FIB stuff
          000
1109 00582 1FF5    DPS024  D1=(5)  =PWROFS         Point at <curs on>/CR/LF/<c off>
          500
1110 00589 8E00            GOSUBL  =RCLSTA         Swap display status into ST
          00
1111 0058F 860             ?ST=0   =Clear          Display-clear bit set?
1112 00592 50              GOYES   DPS030          No.
1113 00594 177             D1=D1+  2*4             Yes.  Point past <curs on>/CR/LF.
1114 00597 8E00    DPS030  GOSUBL  =BF2DSP
          00
1115 0059D 718E    DPS035  GOSUB   fpoll
1116 005A1 CF              CON(2)  =pPWROF
1117 005A3 31DC            LC(2)   =flTNOF
1118 005A7 782F            GOSUB   sflags          Set turnoff flag
1119 005AB 31EC            LC(2)   =flMKOF
1120 005AF 791F            GOSUB   sflagc          Clear makeoff flag
1121 005B3 795F            GOSUB   ATNCL1          Clear ATTN flag
1122 005B7 1C0             D1=D1-  1               Point at ATNDIS
1123 005BA 149             DAT1=A  B               Clear ATTN disable(and ATNFLG)
1124 005BD 1EFF            D1=(4)  =DD1CTL         Disp Control Nibble
          3E
1125 005C3 1590            DAT1=A  1               Turn off display
1126 005C7 1E20            D1=(4)  =ANNAD2         Point to FG ann nibble
          1E
1127 005CD 1590            DAT1=A  1               Reset FG status
1128 005D1 1DFF            D1=(2)  =DD3CTL
1129 005D5 1590            DAT1=A  1               Turn off TIMER #3
1130 005D9 8E00            GOSUBL  =OUT=1          Set OUT=001. (Bug#39-1000(9))
          00
1131 005DF 807             SHUTDN                  Goodnight
1132                 ********************************************************************
1133                 ********************************************************************
```

```
1134                **
1135                ** Name:    SHUTDN  -  Address following DSLEEP shutdown
1136                **
1137                ** Category:   SYSTEM
1138                **
1139                ** Purpose:
1140                **      This is where the CPU will continue executing after
1141                **      being awakened from SHUTDN in deep sleep.
1142                **
1143                **      Hard-configured takeover ROMs should expect to receive
1144                **      control at this address.
1145                **
1146                ** History:
1147                **
1148                **    Date      Programmer              Modification
1149                **  --------    ----------    ----------------------------------
1150                **  11/01/83    B.S.          Added documentation
1151                **
1152                ********************************************************************
1153                ********************************************************************
1154 005E2          =SHUTDN                           Good morning
1155 005E2 8F00     DPS040 GOSBVL =PWCONF             Configure
           000
1156 005E9 20               P=     0
1157 005EB 3290             LC(3)  =bECOMD
           8
1158 005F0 8F00             GOSBVL =I/ODAL            Deallocate External CMD buffer
           000
1159 005F7 8F00             GOSBVL =ALMSRV            Check alarm clock system
           000
1160 005FE 7E0F             GOSUB  ATNCL1             Check ATTN flag
1161 00602 533              GONC   DPS200             Go if was hit.
1162 00605 86D              ?ST=0  13                 Not hit.  Program running?
1163 00608 41               GOYES  DPS150             No.  Poll.
1164 0060A 843              ST=0   3                  Clear timeout bit
1165 0060D 09               C=ST                      Look at alarm bits
1166 0060F 90A              ?C=0   P                  Yes.  On-timer #1,2 or 3 pending?
1167 00612 A0               GOYES  DPS150             No.  Poll.
1168 00614 31DC             LC(2)  =flTNOF            Yes
1169 00618 70BE             GOSUB  sflagc             Clear turnoff flag
1170 0061C 720E     DPS150  GOSUB  fpoll              Deepsleep wake w/o key poll
1171 00620 EF               CON(2) =pDSWNK
1172 00622 20               P=     0
1173 00624 31DC             LC(2)  =flTNOF
1174 00628 786B             GOSUB  sflag?             Has turnoff flag been cleared?
1175 0062C 590              GONC   DPS200             Yes.
1176 0062F 7DDE             GOSUB  ATNCL1             No. Check ATTN flag.
1177 00633 492              GOC    DPS100             Go if hasn't been set.
1178 00636 8E00     DPS200  GOSUBL =nokeys            Flush key buffer.
           00
1179 0063C 20               P=     0
1180 0063E 314C             LC(2)  =flALRM
1181 00642 768E             GOSUB  sflagc             Clear ALARM annunciator
1182 00646 78DD             GOSUB  fpoll              Wakeup poll
1183 0064A DF               CON(2) =pDSWKY
```

```
1184 0064C 8F00          GOSBVL =LOCKD?        Password processing if f1TNOF on
           000
1185 00653 5D0           GONC   DPS101         Go if failed to unlock machine
1186 00656 8D00  acbat?  GOVLNG =ACBAT?        Check AC/BAT and RTNCC
           000
1187 0065D 6F3F  DPS100  GOTO   DPS035         Inrange shortjump from above
1188 00661 602F  DPS101  GOTO   DPS024
1189               ****************************************************************
1190               ****************************************************************
1191               **
1192               ** Name:(S) pPWROF  -  Poll when powering off
1193               **
1194               ** Category:   POLL
1195               **
1196               ** Type:       FPOLL
1197               **
1198               ** Purpose:
1199               **      Poll on entry to deep sleep.
1200               **
1201               ** Should poll be "Handled" (return with XM=0)?:
1202               **      No.
1203               **
1204               ** Meaning of "Handling" Poll (what does code do if handled?):
1205               **      N/A
1206               **
1207               ** Entry conditions for handler (registers, ST, RAM, etc.):
1208               **      Carry set.
1209               **      B[A] = Poll number = pPWROF
1210               **      HEX mode.
1211               **      P=0.
1212               **      f1PWDN set iff deepsleep was called from PWROFF.
1213               **
1214               ** Normal exit conditions from handler if handled (ST, RAM,
1215               ** registers, etc.):
1216               **      N/A
1217               **
1218               ** Normal exit conditions from handler if not handled (ST, RAM,
1219               ** registers, etc.):
1220               **      HEX mode.
1221               **      XM=1.
1222               **
1223               ** Available subroutine levels:  3
1224               **
1225               ** NOTE:
1226               **      The flag f1PWDN indicates that the machine was called
1227               **      from PWROFF, as opposed to CALC mode, programmatic BYE,
1228               **      or somebody else.
1229               **
1230               **      This is a good time to take over.
1231               **
1232               ** What registers/RAM may be used if handled?:
1233               **      N/A
1234               **
1235               ** What registers/RAM may be used if not handled?:
1236               **      All CPU registers except D[A].
```

```
1237            **      All scratch RAM.
1238            **
1239            ** Special memory/pointer considerations (are pointers funny?):
1240            **      May be in CALC mode.
1241            **
1242            ** Envisioned application(s):
1243            **      Some sort of takeover on shutdown.
1244            **
1245            **      Pocket secretary processing alarms at shutdown.
1246            **      Suggested method if an alarm is due and you want to
1247            **      process it at power-off:
1248            **        Schedule immediate wakeup through external alarm.
1249            **        Create external command buffer at wakeup poll using
1250            **          the pocket secretary's handy ACKNOWLEDGE keyword.
1251            **
1252            ** History:
1253            **
1254            **    Date     Programmer             Modification
1255            **  --------   ----------   ----------------------------------
1256            **  03/24/83   NM           Added documentation
1257            **
1258            ********************************************************************
1259            ********************************************************************
1260            ********************************************************************
1261            ********************************************************************
1262            **
1263            ** Name:(S) pDSWNK  -  Poll to awake machine w/o key
1264            **
1265            ** Category:   POLL
1266            **
1267            ** Type:       FPOLL
1268            **
1269            ** Purpose:
1270            **      Poll if machine awoke without ATTN being hit or
1271            **      ON TIMER going off.
1272            **
1273            ** Should poll be "Handled" (return with XM=0)?:
1274            **      No.  I don't think so.
1275            **
1276            ** Meaning of "Handling" Poll (what does code do if handled?):
1277            **      N/A
1278            **
1279            ** Entry conditions for handler (registers, ST, RAM, etc.):
1280            **      Carry set.
1281            **      B[A] = Poll number.
1282            **      HEX mode.
1283            **      P=0.
1284            **
1285            ** Normal exit conditions from handler if handled (ST, RAM,
1286            ** registers, etc.):
1287            **      N/A
1288            **
1289            ** Normal exit conditions from handler if not handled (ST, RAM,
1290            ** registers, etc.):
1291            **      HEX mode.
```

```
1292            **      XM=1.
1293            **      If f1TNOF is cleared during this poll, the machine will
1294            **        wake up AND will circumvent password processing
1295            **        (asking for password if one exists).  If you wish to
1296            **        wake up the machine this way but not give control to
1297            **        the user, setting f1MKOF will force machine back to
1298            **        sleep as soon as it hits the main loop.  This is a
1299            **        way to wake up to process alarms and then return to
1300            **        sleep.
1301            **      If ATNFLG is set during this poll, the machine will
1302            **        continue as though ATTN had been hit... wake up,
1303            **        perform password processing, etc.
1304            **
1305            ** Available subroutine levels:  3
1306            **
1307            ** NOTE:
1308            **      The flag f1PWDN indicates that the machine was called
1309            **      from PWROFF, as opposed to CALC mode, programmatic BYE,
1310            **      or somebody else.  The importance of this is that on
1311            **      return from DSLEEP, PWROFF will recognize and process
1312            **      an external command buffer.  Nobody else will.  So if
1313            **      you wish to create a command buffer to be executed,
1314            **      f1PWDN indicates whether or not it will be ignored.
1315            **
1316            **      The external command buffer was deallocated before the
1317            **      wakeup polls.  If it currently exists, it means that a
1318            **      poll handler has created it.  Think real hard about
1319            **      how badly you want to wipe out somebody else's command.
1320            **      On the other hand, some externally implemented sort of
1321            **      STARTUP may grab this buffer every time.  Such are the
1322            **      dangers in this zoo.  I guess this means not to assume
1323            **      that creating this buffer guarantees that it will be
1324            **      used.
1325            **
1326            ** What registers/RAM may be used if handled?:
1327            **      N/A
1328            **
1329            ** What registers/RAM may be used if not handled?:
1330            **      All registers except D[A].
1331            **      All scratch RAM.
1332            **
1333            ** Special memory/pointer considerations (are pointers funny?):
1334            **      May be in CALC mode.
1335            **
1336            ** Envisioned application(s):
1337            **      Allowing non-ATTN, non-ON-TIMER to awake machine.
1338            **
1339            ** History:
1340            **
1341            **    Date     Programmer             Modification
1342            **  --------   ----------   ----------------------------------
1343            **  03/24/83   NM           Added documentation
1344            **
1345            ****************************************************************
1346            ****************************************************************
```

```
1347            ****************************************************************
1348            ****************************************************************
1349            **
1350            ** Name:(S) pDSWKY  -  Poll if machine wants to wake up
1351            **
1352            ** Category:   POLL
1353            **
1354            ** Type:       FPOLL
1355            **
1356            ** Purpose:
1357            **      Poll if we are going to wake up because:
1358            **        ATTN key was hit.
1359            **        ON TIMER went off.
1360            **        Responder to pDSWNK told us to wake up.
1361            **
1362            ** Should poll be "Handled" (return with XM=0)?:
1363            **      No.
1364            **
1365            ** Meaning of "Handling" Poll (what does code do if handled?):
1366            **      N/A
1367            **
1368            ** Entry conditions for handler (registers, ST, RAM, etc.):
1369            **      Carry set.
1370            **      B[A] = Poll number.
1371            **      HEX mode.
1372            **      P=0.
1373            **
1374            ** Normal exit conditions from handler if handled (ST, RAM,
1375            ** registers, etc.):
1376            **      N/A
1377            **
1378            ** Normal exit conditions from handler if not handled (ST, RAM,
1379            ** registers, etc.):
1380            **      HEX mode.
1381            **      XM=1.
1382            **      If flTNOF cleared, we will wake up without password
1383            **        processing.
1384            **
1385            ** Available subroutine levels:  3
1386            **
1387            ** NOTE:
1388            **      At this point, we are committed to trying to wake up
1389            **      machine.  If, however, flTNOF is set on termination of
1390            **      this poll (it may or may not be set before poll), we
1391            **      will go through password processing... soliciting a
1392            **      password from the user if the machine has been locked.
1393            **
1394            **      The flALRM flag (ALARM annunciator) was cleared just
1395            **      before the poll.  This is the time to set the flag if
1396            **      that annunciator should be on.
1397            **
1398            **      The flag flPWDN indicates that the machine was called
1399            **      from PWROFF, as opposed to CALC mode, programmatic BYE,
1400            **      or somebody else.  The importance of this is that on
1401            **      return from DSLEEP, PWROFF will recognize and process
```

```
1402            **      an external command buffer.  Nobody else will.  So if
1403            **      you wish to create a command buffer to be executed,
1404            **      fIPWDN indicates whether or not it will be ignored.
1405            **
1406            **      The external command buffer was deallocated before the
1407            **      wakeup polls.  If it currently exists, it means that a
1408            **      poll handler has created it.  Think real hard about
1409            **      how badly you want to wipe out somebody else's command.
1410            **      On the other hand, some externally implemented sort of
1411            **      STARTUP may grab this buffer every time.  Such are the
1412            **      dangers in this zoo.  I guess this means not to assume
1413            **      that creating this buffer guarantees that it will be
1414            **      used.
1415            **
1416            ** What registers/RAM may be used if handled?:
1417            **      N/A
1418            **
1419            ** What registers/RAM may be used if not handled?:
1420            **      All CPU registers except D[A].
1421            **      All scratch RAM.
1422            **
1423            ** Special memory/pointer considerations (are pointers funny?):
1424            **      May be in CALC mode.
1425            **
1426            ** Envisioned application(s):
1427            **      Takeover ROM at powerup.
1428            **
1429            **      Alarm processing.
1430            **
1431            ** History:
1432            **
1433            **    Date     Programmer             Modification
1434            **  --------   ----------   ------------------------------------
1435            **  10/25/83   NM           Updated documentation
1436            **
1437            ********************************************************************
1438            ********************************************************************
```

```
1439                    STITLE LCDINI - Initialize LCD
1440                ******************************************************************
1441                ******************************************************************
1442                **
1443                ** Name:(S) LCDINI  -  Initialize LCD display
1444                **
1445                ** Category:   DSPUTL
1446                **
1447                ** Purpose:
1448                **      Initialize LCD row driver and contrast, turn display on
1449                **
1450                ** Entry:
1451                **      P      =  0
1452                **
1453                ** Exit:
1454                **      P      =  0
1455                **      Carry clear
1456                **
1457                ** Calls:      None
1458                **
1459                ** Uses.......
1460                **  Inclusive: C(W)
1461                **
1462                ** Stk lvls:   0
1463                **
1464                ** History:
1465                **
1466                **    Date     Programmer              Modification
1467                **  --------   ----------   ----------------------------------
1468                **  10/25/83   B.S.         Added documentation
1469                **
1470                ******************************************************************
1471                ******************************************************************
1472 00665 1B05  =LCDINI DO=(5) =ROWDVR
           3E2
1473 0066C 3F80         LCHEX  8001400220041008   Bit pattern for driving
           0140
           0220
           0410
           08
1474                                                  display
1475 0067E 1547         DATO=C W                Write out display drivers
1476 00682 19EF         DO=(2) (=DD1CTL)-1      Display contrast/control byte
1477 00686 3191         LCHEX  19               Set contrast to 9, display on
1478 0068A 14C          DATO=C B                Turn on display, Set contrast
1479 0068D 03    RTNCC  RTNCC
1480                *-
1481                *-
1482 0068F B1    =COLDSM NIBHEX B1               Escape
1483 00691 C3D4         NIBASC \<Memory\
           56D6
           F627
           97
1484 0069F 02C4         NIBASC \ Lost\
           F637
```

```
                47
1485 006A9 D0A0         NIBHEX D0A0FF          CR/LF/eol
                FF
```

```
1486                    STITLE TNOFF?
1487                ****************************************************************
1488                ****************************************************************
1489                **
1490                ** Name:    TNOFF?  -  Check for flTNOF and flMKOF
1491                **
1492                ** Category:   GENUTL
1493                **
1494                ** Purpose:
1495                **      Determine if machine should go to sleep.
1496                **
1497                ** Entry:
1498                **      P=0.
1499                **
1500                ** Exit:
1501                **      P=0.
1502                **      Carry set if machine should go to sleep.  The check
1503                **        in the main loop and in calc mode.  Carry set
1504                **        typically means that user tried to jump password
1505                **        with INIT 1.
1506                **
1507                ** Calls:      SFLAG?
1508                **
1509                ** Uses.......
1510                **             A(A),C(S),C(5-0),D(A)
1511                **
1512                ** Stk lvls:   2
1513                **
1514                ** History:
1515                **
1516                **    Date      Programmer              Modification
1517                **  --------    ----------   ----------------------------------
1518                **  06/27/83    NM           Stripped out of MAINLP code
1519                **
1520                ****************************************************************
1521                ****************************************************************
1522 006AF 31EC =TNOFF? LC(2)  =flMKOF
1523 006B3 7DDA         GOSUB  sflag?          Is Makeoff flag set?
1524 006B7 400          RTNC                   Yes.  Must power down.
1525 006BA 31DC         LC(2)  =flTNOF
1526 006BE 65DA         GOTO   sflag?          Check turnoff flag
```

```
1527                    STITLE SLEEP
1528                ********************************************************************
1529                ********************************************************************
1530                **
1531                ** Name:(S) SLEEP   -  Scan KB, do LSLEEP if key buffer empty
1532                ** Name:(S) LSLEEP  -  Light Sleep
1533                **
1534                ** Category:   GENUTL
1535                **
1536                ** Purpose:
1537                **      SLEEP:
1538                **        Debounces keyboard and shuts CPU down unless keys are
1539                **        in buffer or down.
1540                **      LSLEEP:
1541                **        Shuts CPU down (enters low power state) until some
1542                **        activity on the bus or the keyboard wakes up CPU.
1543                **
1544                ** Entry:
1545                **
1546                ** Exit:
1547                **      P      = 0
1548                **      Carry clear if keys in buffer
1549                **      Carry set if no keys were in buffer
1550                **
1551                ** Calls:      DEBNCE,KEY?
1552                **
1553                ** Uses.......
1554                **  Exclusive: C(A)
1555                **  Inclusive: A(W),B(W),C(W),DO
1556                **
1557                ** Stk lvls:   1
1558                **
1559                ** Algorithm:
1560                **      Debounce for 8/512ths second then scan keyboard
1561                **      If key buffer not empty
1562                **           then return with carry clear
1563                **      If any keys are down
1564                **           then return with carry set
1565                **      Shut down CPU
1566                **      If MP=1 or flMPI set then
1567                **        Go to MPI
1568                **      Return with carry set
1569                **
1570                ** History:
1571                **
1572                **    Date     Programmer              Modification
1573                **  --------   ----------   ------------------------------------
1574                **  07/15/82   B.S.         Updated documentation
1575                **
1576                ********************************************************************
1577                ********************************************************************
1578 006C2 7000   =SLEEP  GOSUB  =DEBNCE        Scan keyboard
1579 006C6 7840          GOSUB  KEY?           Is the key buffer empty?
1580 006CA 500           RTNNC                 Return if buffer not empty
1581 006CD 8E00   =LSLEEP GOSUBL =OUT=F         Set OUT=00F.  (Bug#39-1000(9))
```

```
            00
1582                *
1583 006D3 D2              C=0    A
1584 006D5 803             C=IN
1585 006D8 8AE             ?C#0   A
1586 006DB 00              RTNYES
1587                *
1588 006DD 7220            GOSUB  RCflMP
1589 006E1 841             ST=0   1
1590 006E4 0B              CSTEX
1591 006E6 15C0            DAT0=C 1                Clear flMPI
1592 006EA 807             SHUTDN
1593 006ED 838             ?MP=0                   Module pulled?
1594 006F0 60              GOYES  LSLE10           Not lately.  Detected by intrpt?
1595 006F2 68AA  LSLE05    GOTO   MPI              Yes.  Handle it.
1596 006F6 7900  LSLE10    GOSUB  RCflMP
1597 006FA 871             ?ST=1  1                flMPI set?
1598 006FD 5F              GOYES  LSLE05           Yes.  MPI processing.
1599 006FF 0B              CSTEX                   No.  Restore status bits.
1600 00701 02              RTNSC                   All is well.
1601                *
1602 00703 1B3E  RCflMP    DO=(5) (=SYSFLG)+10
             6F2
1603 0070A 15E0            C=DAT0 1
1604 0070E 0B              CSTEX
1605 00710 01              RTN
1606                *
```

```
1607                    STITLE KEY?
1608                ********************************************************************
1609                ********************************************************************
1610                **
1611                ** Name:    KEY?    -  Is key buffer empty?
1612                **
1613                ** Category:   KEYUTL
1614                **
1615                ** Purpose:
1616                **      Test if key buffer is empty.
1617                **
1618                ** Entry:
1619                **
1620                ** Exit:
1621                **      Carry set iff keybuffer empty.
1622                **
1623                ** Calls:      None.
1624                **
1625                ** Uses.......
1626                **              A[XS],D0.
1627                **
1628                ** Stk lvls:   0
1629                **
1630                ** History:
1631                **
1632                **    Date      Programmer              Modification
1633                **  --------    ----------   ------------------------------------
1634                **  11/10/82    NM           Added documentation
1635                **
1636                ********************************************************************
1637                ********************************************************************
1638 00712 1B34 =KEY?   D0=(5) =KEYPTR
           4F2
1639 00719 1522          A=DAT0 XS
1640 0071D 630E          GOTO   ATNCL2          Is buffer empty?
```

```
1641                    STITLE CKSREQ
1642                ********************************************************************
1643                ********************************************************************
1644                **
1645                ** Name:(S) CKSREQ  -  Handle service requests
1646                **
1647                ** Category:   GENUTL
1648                **
1649                ** Purpose:
1650                **      Handle service requests.  This routine recognizes
1651                **      several possible sources of service requests:
1652                **        1) Timer 1--Display code needs service.
1653                **        2) Timer 2--Clock system needs service.
1654                **        3) Timer 3--Battery check code needs service.
1655                **
1656                **      After examining above, CKSREQ performs a poll which
1657                **      allows:
1658                **        1) Handling of SREQs we don't recognize.
1659                **        2) Handling related to recognized SREQs (e.g.,
1660                **           scheduling a new external alarm through clock
1661                **           system).
1662                **
1663                **      This code is typically called when:
1664                **        1) We wake up from a sleep state (delay, etc.).
1665                **        2) We recognize that an SREQ is exerted at certain
1666                **           points in the mainframe (e.g., interpreter loop).
1667                **
1668                ** Entry:
1669                **      Hex Mode
1670                **
1671                ** Exit:
1672                **      Hex Mode
1673                **
1674                ** Calls:      ACBTSR,CKTMOU,DSPUPD,ALMSRV,PUTPND,FPOLL
1675                **
1676                ** Uses.......
1677                **             A,B,C,D,P,D0,D1,32 nibs at SCRTCH
1678                **
1679                ** Stk lvls:   4
1680                **
1681                ** NOTE:
1682                **      This code saves the status bits in the user-status
1683                **      save area used by the display code.
1684                **
1685                ** Algorithm:
1686                **      Set BAT annunciator if low battery
1687                **      Save caller's status bits in display status area
1688                **      If display timer has timed out
1689                **        then update display (blink cursor, etc.)
1690                **      Check alarm clock system
1691                **      Clear external alarm bit in clock system status
1692                **      If Except bit set or service request still pending then
1693                **        Poll (pSREQ)
1694                **      Restore caller's status
1695                **      Return
```

```
1696                 **
1697                 ** History:
1698                 **
1699                 **    Date     Programmer          Modification
1700                 **  --------   ----------  ----------------------------------
1701                 **  02/25/83   NM          Added documentation
1702                 **  10/25/83   B.S.        Updated documentation
1703                 **
1704                 ***************************************************************
1705                 ***************************************************************
1706 00721 8F00 =CKSREQ GOSBVL =ACBTSR        Chk Bat if timer3 pending
             000
1707 00728 8E00         GOSUBL =GETSTA        Get display status
             00
1708 0072E 1BDF         DO=(5) (=TIMER1)+5
             3E2
1709 00735 8F00         GOSBVL =CKTMOU        Time for display update?
             000
1710 0073C 580          GONC   CKSR10         No, then continue
1711 0073F 8E00         GOSUBL =DSPUP1        Yes, then update display
             00
1712 00745       CKSR10
1713                 *
1714                 * Alarm clock system does very little processing if its timer
1715                 * is not timed out
1716                 *
1717 00745 8F00         GOSBVL =ALMSRV        Service alarm clock system
             000
1718 0074C 845          ST=0   5              Clear external alarm bit
1719 0074F 09           C=ST
1720 00751 8F00         GOSBVL =PUTPND        Write out pending alarm bitmap
             000
1721                 *
1722 00758 87C          ?ST=1  Except         Exception flag set?
1723 0075B D0           GOYES  CKSR20         Yes, must poll.
1724 0075D 80E          SREQ?                 SREQ still pending?
1725 00760 834          ?SR=0
1726 00763 B0           GOYES  CKSR30         No.
1727 00765 824          SR=0                  Yes, then clear bit
1728 00768 76BC  CKSR20 GOSUB  fpoll          Poll.
1729 0076C 9F           CON(2) =pSREQ
1730 0076E 8C00  CKSR30 GOLONG =USRSTA
             00
1731                 *
1732                 ***************************************************************
1733                 ***************************************************************
1734                 **
1735                 ** Name:(S) pSREQ   -  Service Request poll
1736                 **
1737                 ** Category:   POLL
1738                 **
1739                 ** Type:       FPOLL
1740                 **
1741                 ** Purpose:
1742                 **      Allow LEXFILE processing when a hardware service
```

```
1743            **      request is exerted.
1744            **
1745            ** Should poll be "Handled" (return with XM=0)?:
1746            **      NO!!  NEVER!!
1747            **
1748            ** Meaning of "Handling" Poll (what does code do if handled?):
1749            **      N/A
1750            **
1751            ** Entry conditions for handler (registers, ST, RAM, etc.):
1752            **      Carry set.
1753            **      B[A] = pSREQ.
1754            **      HEX mode.
1755            **      P=0.
1756            **      flDORM flag is set if machine is in main loop
1757            **        (dormant).
1758            **
1759            ** Normal exit conditions from handler if handled (ST, RAM,
1760            ** registers, etc.):
1761            **      N/A
1762            **
1763            ** Normal exit conditions from handler if not handled (ST, RAM,
1764            ** registers, etc.):
1765            **      HEX mode.
1766            **      XM=1.
1767            **
1768            ** Available subroutine levels:
1769            **      2
1770            **
1771            ** NOTE:
1772            **      D[A], and R0-R4 must be preserved.
1773            **
1774            **      A copy of the user's status bits as they existed on
1775            **      entry to CKSREQ exists at DSPSTA (the 3 nibbles used
1776            **      by display routines to save status bits).  Do not
1777            **      destroy this copy; it is needed so ST can be restored
1778            **      after the poll.
1779            **
1780            **      The available scratch RAM is, conveniently, just enough
1781            **      to use the clock system safely.  You can save R0 and R1
1782            **      at SCRTCH, D[A] at SCREX0, and subroutine levels in
1783            **      SCREX1, SCREX2, SCREX3.
1784            **
1785            **      This poll IS NOT a time to take over the machine.  It
1786            **      may occur during display delay, program execution,
1787            **      character editing, wait, etc.  This poll IS a time for
1788            **      handling service requests non-disruptively (such as
1789            **      scheduling an alarm, doing a beep, setting the
1790            **      exception flag, or anything else which does not disrupt
1791            **      the flow of whatever was going on when you generated
1792            **      your service request) and for setting up to take over
1793            **      the machine (such as setting a flag which tells you to
1794            **      grab the exception poll or the deepsleep poll).
1795            **
1796            ** What registers/RAM may be used if handled?:
1797            **      N/A
```

```
1798            **
1799            ** What registers/RAM may be used if not handled?:
1800            **      A-C, D[15-5], D0, D1, P, ST.
1801            **      First 32 nibbles at SCRTCH.
1802            **      SCREX0, SCREX1, SCREX2, SCREX3.
1803            **
1804            ** Special memory/pointer considerations (are pointers funny?):
1805            **      We could be in CALC mode.
1806            **
1807            ** Envisioned application(s):
1808            **      Scheduling external alarms though the clock system is
1809            **      one very important application.  If a few simple rules
1810            **      are followed when dealing with the clock system,
1811            **      everything should work just fine:
1812            **
1813            **        Rule #1: If the current external alarm is past due
1814            **                 (before current time), you may schedule an
1815            **                 external alarm.
1816            **        Rule #2: If the current external alarm is not past
1817            **                 due, you may only schedule an external alarm
1818            **                 if a) your alarm is not past due, and b) it
1819            **                 occurs before the currently scheduled
1820            **                 external alarm.
1821            **        Rule #3: You can tell if one of your alarms is
1822            **                 pending by comparing it to the current time.
1823            **                 Do not count on the current value in the
1824            **                 external alarm slot being yours... somebody
1825            **                 may have followed rule #2 and jumped in
1826            **                 ahead of you.
1827            **
1828            **
1829            **      Another application: Remote Keyboard.  Presumably your
1830            **      code is associated with some hardware (an HPIL mailbox,
1831            **      maybe) which has exerted a service request because of a
1832            **      remote keyboard.  Take this poll as an opportunity to
1833            **      stuff a key# in the keybuffer.  If it is not a key#
1834            **      which can be understood by the machine, you can define
1835            **      it by handling the key definition poll.
1836            **
1837            ** History:
1838            **
1839            **    Date     Programmer            Modification
1840            **  --------   ----------   ----------------------------------
1841            **  03/23/83   NM           Added documentation
1842            **
1843            ********************************************************************
1844            ********************************************************************
```

```
1845                    STITLE KYDN?
1846                ******************************************************************
1847                ******************************************************************
1848                **
1849                ** Name:(S) KYDN?   -  Is a Key Down in Current Row?
1850                **
1851                ** Category:   CONFIG
1852                **
1853                ** Purpose:
1854                **      Determine if a key is down which could cause a problem
1855                **      for configuration.
1856                **
1857                ** Entry:
1858                **
1859                ** Exit:
1860                **      Carry clear if a key is down in the currently energized
1861                **      row(s), else carry clear.
1862                **
1863                ** Calls:      None.
1864                **
1865                ** Uses.......
1866                **             None.
1867                **
1868                ** Stk lvls:   1
1869                **
1870                ** NOTE:
1871                **      A brief description of the problem: If 2 or more keys
1872                **      are down in a column, and a row containing one of the
1873                **      keys is energized, the multiple keys short the rows
1874                **      together, resulting in energizing multiple rows.  In
1875                **      configuration, this amounts to addressing more than one
1876                **      port daisy chain at once, which can lead to disaster.
1877                **      This routine is called at appropriate times to ensure
1878                **      that no keys are down that can screw up configuration.
1879                **
1880                ** Detail:
1881                **      Preserves all registers at the expense of a subroutine
1882                **      level.
1883                **
1884                ** History:
1885                **
1886                **    Date     Programmer             Modification
1887                **  --------   ----------   ----------------------------------
1888                **  09/16/82   NM           Added documentation
1889                **
1890                ******************************************************************
1891                ******************************************************************
1892 00774 06     =KYDN?  RSTK=C                  Save C
1893 00776 803          C=IN
1894 00779 F2           CSL    A
1895 0077B C6           C=C+C  A                 Knock off upper 2 bits in IN reg
1896 0077D C6           C=C+C  A                 so we are left with lower 14.
1897 0077F CE           C=C-1  A                 Set carry if no keys down (IN=0)
1898 00781 07           C=RSTK                   Restore C
1899 00783 01           RTN
```

```
1900 00785              END
```

```
 ACBAT?  Ext                 -  1186
 ACBTSR  Ext                 -  1706
 ALMSRV  Ext                 -  1159  1717
 ANNAD2  Abs  188674 #2E102  -    13   231  1126
 ATNCL1  Abs    1296 #00510  -   913  1121  1160  1176
 ATNCL2  Abs    1313 #00521  -   917  1640
=ATNCLR  Abs    1296 #00510  -   912   703   875
 ATNFLG  Abs  193602 #2F442  -    13   913
 ATNTST  Abs    1242 #004DA  -   866   663   668   674
 ATTN10  Abs    1275 #004FB  -   873   871
=ATTNTN  Abs    1251 #004E3  -   868   708   716
 AUTCLR  Ext                 -   386   874
 AUTOCK  Ext                 -   578
 AUTXQ7  Ext                 -   580
 AVMEMS  Abs  193940 #2F594  -    13   381
 BF2DPP  Ext                 -   669
 BF2DSP  Ext                 -   404  1114
 BLDBIT  Ext                 -   263
 BLDDSP  Ext                 -   264
 BLDLCD  Ext                 -   265
 C=RAME  Ext                 -   380
 C=SCRL  Ext                 -   664
 CALC    Ext                 -   830
=CALCJ   Abs    1217 #004C1  -   830   717
 CHEDIT  Ext                 -   266
 CK"ON"  Ext                 -   866
 CKSR10  Abs    1861 #00745  -  1712  1710
 CKSR20  Abs    1896 #00768  -  1728  1723
 CKSR30  Abs    1902 #0076E  -  1730  1726
=CKSREQ  Abs    1825 #00721  -  1706
=CKSUM1  Abs      13 #0000D  -    81
 CKTMOU  Ext                 -  1709
 CLCERR  Ext                 -   655
 CLCOLL  Ext                 -   677
=CLDST1  Abs     494 #001EE  -   343    78
 CLDST2  Abs     539 #0021B  -   356
 CMDOFF  Abs    1224 #004C8  -   833   700   775
 CMDSTK  Ext                 -   813
 CMDSTk  Abs    1183 #0049F  -   813   719
 CMOSTV  Abs    5775 #0168F  -    13   206   345
 CMOSTW  Abs  193592 #2F438  -    13   204   344
=COLDSM  Abs    1679 #0068F  -  1482   403
=COLDST  Abs       0 #00000  -    77   210
 COLLAP  Ext                 -   676
 CONF    Ext                 -   369
 CONTK   Ext                 -   824
 CONTKJ  Abs    1205 #004B5  -   824   710
=CRLFPR  Abs     808 #00328  -   648
 CSTRST  Ext                 -   379
 CURSBJ  Ext                 -   810
 CURSDJ  Ext                 -   808
 CURSFR  Ext                 -   704
 CURSTJ  Ext                 -   809
 CURSUJ  Ext                 -   807
 Clear   Ext                 -   870  1111
```

DD1CTL	Abs	189439	#2E3FF	-	13	1124	1476
DD3CTL	Abs	188927	#2E1FF	-	13	1128	
DEBNCE	Ext			-	267	1578	
DISINT	Abs	193648	#2F470	-	13	197	
DPS010	Abs	1401	#00579	-	1106	959	
DPS023	Abs	1403	#0057B	-	1108		
DPS024	Abs	1410	#00582	-	1109	1188	
DPS030	Abs	1431	#00597	-	1114	1112	
DPS035	Abs	1437	#0059D	-	1115	1187	
DPS040	Abs	1506	#005E2	-	1155		
DPS100	Abs	1629	#0065D	-	1187	1177	
DPS101	Abs	1633	#00661	-	1188	1185	
DPS150	Abs	1564	#0061C	-	1170	1163	1167
DPS200	Abs	1590	#00636	-	1178	1161	1175
=DSLEEP	Abs	1389	#0056D	-	1102		
DSPCHA	Ext			-	268		
DSPRST	Ext			-	269		
DSPUP1	Ext			-	1711		
DSPUPD	Ext			-	270		
DWIDTH	Abs	194895	#2F94F	-	13	398	401
EDITWF	Ext			-	407		
EOLk	Abs	1193	#004A9	-	817	707	
Except	Abs	12	#0000C	-	12	1722	
FIBOFF	Ext			-	1108		
FINLIN	Ext			-	872		
FPoll	Ext			-	722		
GETSTA	Ext			-	869	1707	
I/OALL	Ext			-	417	782	
I/ODAL	Ext			-	683	1158	
I/OFND	Ext			-	961	966	
IEXKEY	Abs	1064	#00428	-	773	688	
IEXKYE	Abs	1150	#0047E	-	802	784	
INITCL	Ext			-	372		
INTA	Abs	193552	#2F410	-	13	167	255
INTB	Abs	193568	#2F420	-	13	169	252
INTM	Abs	193584	#2F430	-	13	242	
INTR10	Abs	68	#00044	-	178	176	
INTR19	Abs	98	#00062	-	190	189	
INTR20	Abs	118	#00076	-	196	186	194
INTR26	Abs	145	#00091	-	204	200	
INTR27	Abs	170	#000AA	-	213	209	
INTR28	Abs	185	#000B9	-	228	216	
INTR4	Abs	193536	#2F400	-	13	165	257
=INTR50	Abs	219	#000DB	-	241	203	
INTR90	Abs	241	#000F1	-	249	247	
=INTRPT	Abs	15	#0000F	-	163		
IS-TBL	Abs	194445	#2F78D	-	13	392	
=KEY?	Abs	1810	#00712	-	1638	1579	
KEYPTR	Abs	193603	#2F443	-	13	1638	
KEYSAV	Abs	193634	#2F462	-	13	228	
KEYSCN	Ext			-	271		
=KYDN?	Abs	1908	#00774	-	1892		
=LCDINI	Abs	1637	#00665	-	1472	358	
LINEP	Ext			-	818		
LINEP+	Ext			-	799		

```
 LINEPj  Abs    1144 #00478 -   799   964
 LOCKD?  Ext                -  1184
 LSLE05  Abs    1778 #006F2 -  1595  1598
 LSLE10  Abs    1782 #006F6 -  1596  1594
=LSLEEP  Abs    1741 #006CD -  1581   282
 MAIN01  Abs     778 #0030A -   573   570
=MAIN05  Abs     824 #00338 -   653   579
 MAIN06  Abs     840 #00348 -   657   654
 MAIN08  Abs     888 #00378 -   669   666
=MAIN10  Abs     894 #0037E -   671   662   814
=MAIN30  Abs     894 #0037E -   672
 MAIN40  Abs     991 #003DF -   701   696   699
=MAINLP  Abs     765 #002FD -   568   876   968
 MAKEBF  Ext                -   272
 MAXCMD  Abs  194934 #2F976 -    13   364
 MEMERX  Ext                -   804
 MFERR   Ext                -   337
 MOVEU3  Ext                -   791
 MPI     Abs     411 #0019B -   319   195  1595
 MPI10   Abs     436 #001B4 -   326   334
 MPI20   Abs     466 #001D2 -   335   328
 NOPRGM  Ext                -   657
 NOSCRL  Ext                -   873
 NoCont  Abs      14 #0000E -    12   388   675
 Nxtstm  Ext                -   279
 OUT=1   Ext                -  1130
 OUT=F   Ext                -  1581
 PAS2+   Abs       2 #00002 -    11
 POPBUF  Ext                -   273
=PROMPT  Abs     812 #0032C -   649
 PUTPND  Ext                -  1720
 PWCONF  Ext                -   336  1155
 PWIDTH  Abs  194904 #2F958 -    13   401
 PWR050  Abs    1349 #00545 -   963   967
 PWR100  Abs    1356 #0054C -   965   962
=PWROFF  Abs    1318 #00526 -   955   571   718
=PWROFS  Abs    1375 #0055F -   971  1109
 PgmRun  Abs      13 #0000D -    12   387
 RCLSTA  Ext                -  1110
 RCflMP  Abs    1795 #00703 -  1602   190  1588  1596
 RNSEED  Abs  194302 #2F6FE -    13   423
 ROWDVR  Abs  189264 #2E350 -    13  1472
 RPTKY   Ext                -   274
 RTNCC   Abs    1677 #0068D -  1479
 RUNK    Ext                -   821
 RUNKj   Abs    1199 #004AF -   821   709
 Range   Ext                -   698
 SCRLLR  Ext                -   275
 SCROLT  Abs  194886 #2F946 -    13   361
 SFLAG?  Ext                -   283
 SFLAGC  Ext                -   834
 SFLAGS  Ext                -   835
=SHUTDN  Abs    1506 #005E2 -  1154
=SLEEP   Abs    1730 #006C2 -  1578
 SST     Ext                -   827
```

```
 SSTj    Abs    1211 #004BB -   827   711
 STMBCL  Ext                -   681
 SYSFLG  Abs  194265 #2F6D9 -    13  1602
 TBLJMC  Ext                -   706
 TIMER1  Abs  189432 #2E3F8 -    13  1708
 TIMER3  Abs  188920 #2E1F8 -    13   323
=TNOFF?  Abs    1711 #006AF -  1522   569
 USRSTA  Ext                -  1730
 VECTOR  Abs  193596 #2F43C -    13   350
=VECTR!  Abs     517 #00205 -   349
 VIEWD1  Ext                -   276
=WAKEUP  Abs     894 #0037E -   673
 WARM1X  Ext                -   281
 WIPOUT  Ext                -   354   385
 WRTTMR  Ext                -   330
 acbat?  Abs    1622 #00656 -  1186   376
 bECOMD  Abs    2057 #00809 -    12   960  1157
 bFIB    Abs    2051 #00803 -    12   413
 bIEXKY  Abs    2050 #00802 -    12   682   781
 bSTART  Abs    2056 #00808 -    12   965
=bldbit  Abs     285 #0011D -   263
=blddsp  Abs     292 #00124 -   264
=bldlcd  Abs     299 #0012B -   265
=chedit  Abs     306 #00132 -   266   687
 clcerr  Abs     833 #00341 -   655   574
 cursbj  Abs    1177 #00499 -   810   715
 cursdj  Abs    1165 #0048D -   808   713
 curstj  Abs    1171 #00493 -   809   714
 cursuj  Abs    1159 #00487 -   807   712
=debnce  Abs     313 #00139 -   267   237
=dspcha  Abs     320 #00140 -   268
=dsprst  Abs     327 #00147 -   269   355
=dspupd  Abs     334 #0014E -   270
 eMPI    Ext                -   339
 fCALC?  Ext                -   573   653
 flALRM  Abs     -60 #FFFC4 -    12  1180
 flCMDS  Abs     -47 #FFFD1 -    12   833
 flDORM  Abs     -43 #FFFD5 -    12   658   685   701   773
 flMKOF  Abs     -50 #FFFCE -    12  1119  1522
 flNOPR  Abs     -26 #FFFE6 -    12   660   679
 flPWDN  Abs     -49 #FFFCF -    12   957  1104
 flTNOF  Abs     -51 #FFFCD -    12  1117  1168  1173  1525
 fpoll   Abs    1058 #00422 -   722   429   575  1115  1170  1182  1728
=keyscn  Abs     341 #00155 -   271   335
=lsleep  Abs     400 #00190 -   282
=makebf  Abs     348 #0015C -   272
 nokeys  Ext                -   868  1178
 pCLDST  Abs     255 #000FF -    12   430
 pDSWKY  Abs     253 #000FD -    12  1183
 pDSWNK  Abs     254 #000FE -    12  1171
 pMNLP   Abs     250 #000FA -    12   576
 pPWROF  Abs     252 #000FC -    12  1116
 pSREQ   Abs     249 #000F9 -    12  1729
=popbuf  Abs     355 #00163 -   273
=reconf  Abs     383 #0017F -   277
```

```
=rptky   Abs     362 #00016A -   274
=scrllr  Abs     369 #00171 -    275   667
=sflag?  Abs     404 #00194 -    283   661  1174  1523  1526
=sflagc  Abs    1228 #004CC -    834   680   702   774  1105  1120  1169  1181
=sflags  Abs    1235 #004D3 -    835   659   686   958  1118
=viewd1  Abs     376 #00178 -    276
=warm1x  Abs     394 #0018A -    281
```

Input Parameters

  Source file name is SB&DVR::MS

  Listing file name is SB/DVR:TI:ML::-1

  Object file name is SB%DVR:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    TTTTT  III   &     RRRR   EEEEE  V   V
 2            *      T     I   & &    R   R  E      V   V
 3            *      T     I   & &    R   R  E      V   V
 4            *      T     I    &     RRRR   EEEE    V V
 5            *      T     I   & & &  R R    E       V V
 6            *      T     I   &  &   R  R   E        V
 7            *      T    III   && &  R   R  EEEEE    V
 8
 9                     TITLE  HP-71 Revision Number
10 00000 24           NIBASC \B\
11 00002              END
```

Input Parameters

  Source file name is TI&REV::MS

  Listing file name is TI/REV:TI:ML::-1

  Object file name is TI%REV:TI:MS::-1

```
                                  111111
                        0123456789012345
```
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS   BBBB    &     BBBB   III  TTTTT
 2            *    S   S  B   B  & &    B   B   I     T
 3            *    S      B   B  & &    B   B   I     T
 4            *     SSS   BBBB    &     BBBB    I     T
 5            *        S  B   B  & & &  B   B   I     T
 6            *    S   S  B   B  &  &   B   B   I     T
 7            *     SSS   BBBB    && &  BBBB   III    T
 8            *
 9                     TITLE  ASCII Tables (Rounded Letters)
10 00787             ABS    #787
11                   ************************************************************
12                   ************************************************************
13                   **
14                   ** Name:(S) ASCII   -  ASCII Bit Pattern Tables
15                   **
16                   ** Category:   DSPUTL
17                   **
18                   ** Purpose:
19                   **      Bit patterns for built in character set.
20                   **
21                   ** Detail:
22                   **      The bit pattern for each character requires 10
23                   **      nibbles.  Each of the 5 pairs of nibbles defines
24                   **      one display column.  Each column has 8 bits where
25                   **      the lsb of the byte is the top row and the msb is
26                   **      the bottom row.  The bit pattern for an ASCII char
27                   **      may be found by reading 10 nibbles at the address
28                   **      ASCII + 10 * (Char#).
29                   **
30                   ** History:
31                   **
32                   **    Date     Programmer             Modification
33                   **  --------   ----------   ----------------------------------
34                   **  07/29/83   B.S.         Updated documentation
35                   **
36                   ************************************************************
37                   ************************************************************
38
39 00787 F7F7 =REPCUR NIBHEX F7F7F7F7F7     Replace cursor
         F7F7
         F7
40                   *              @ @ @ @ @ .
41                   *              @ @ @ @ @ .
42                   *              @ @ @ @ @ .
43                   *              @ @ @ @ @ .
44                   *              @ @ @ @ @ .
45                   *              @ @ @ @ @ .
46                   *              @ @ @ @ @ .
47                   *              . . . . . .
48 00791 80C1 =INSCUR NIBHEX 80C1E3F7C1     Insert Cursor
         E3F7
         C1
49                   *              . . . @ . .
50                   *              . . @ @ . .
51                   *              . @ @ @ @ .
```

```
     52             *              @ @ @ @ @ .
     53             *              . @ @ @ @ .
     54             *              . . @ @ . .
     55             *              . . . @ . .
     56             *              . . . . . .
     57 0079B       =ASCII
     58 0079B 0000          NIBHEX 0000000000     Char 00: Null
              0000
              00
     59             *              . . . . . .
     60             *              . . . . . .
     61             *              . . . . . .
     62             *              . . . . . .
     63             *              . . . . . .
     64             *              . . . . . .
     65             *              . . . . . .
     66             *              . . . . . .
     67 007A5 0070          NIBHEX 0070507000     Char 01: Degree
              5070
              00
     68             *              . @ @ @ . .
     69             *              . @ . @ . .
     70             *              . @ @ @ . .
     71             *              . . . . . .
     72             *              . . . . . .
     73             *              . . . . . .
     74             *              . . . . . .
     75             *              . . . . . .
     76 007AF 5492          NIBHEX 5492119254     Char 02: XBar
              1192
              54
     77             *              @ @ @ @ @ .
     78             *              . . . . . .
     79             *              @ . . . @ .
     80             *              . @ . @ . .
     81             *              . . @ . . .
     82             *              . @ . @ . .
     83             *              @ . . . @ .
     84             *              . . . . . .
     85 007B9 80C1          NIBHEX 80C1A28080     Char 03: Left Arrow
              A280
              80
     86             *              . . . . . .
     87             *              . . @ . . .
     88             *              . @ . . . .
     89             *              @ @ @ @ @ .
     90             *              . @ . . . .
     91             *              . . @ . . .
     92             *              . . . . . .
     93             *              . . . . . .
     94 007C3 8344          NIBHEX 8344448344     Char 04: Alpha
              4483
              44
     95             *              . . . . . .
     96             *              . . . . . .
```

```
 97            *              . @ @ . @ .
 98            *              @ . . @ . .
 99            *              @ . . @ . .
100            *              @ . . @ . .
101            *              . @ @ . @ .
102            *              . . . . . .
103 007CD CFA0         NIBHEX CFA0A4A443     Char 05: Beta
          A4A4
          43
104            *              . . . . . .
105            *              . @ @ @ . .
106            *              @ . . . @ .
107            *              @ @ @ @ . .
108            *              @ . . . @ .
109            *              @ . . . @ .
110            *              @ . @ @ . .
111            *              @ . . . . .
112 007D7 F710         NIBHEX F710101030     Char 06: Gamma
          1010
          30
113            *              @ @ @ @ @ .
114            *              @ . . . @ .
115            *              @ . . . . .
116            *              @ . . . . .
117            *              @ . . . . .
118            *              @ . . . . .
119            *              @ . . . . .
120            *              . . . . . .
121 007E1 03C3         NIBHEX 03C3E7C303     Char 07: Bell
          E7C3
          03
122            *              . . . . . .
123            *              . . @ . . .
124            *              . @ @ @ . .
125            *              . @ @ @ . .
126            *              @ @ @ @ @ .
127            *              @ @ @ @ @ .
128            *              . . @ . . .
129            *              . . . . . .
130 007EB F151         NIBHEX F151A08586     Char 08: Backspace
          A085
          86
131            *              @ @ . . . .
132            *              @ . @ . . .
133            *              @ @ . . . .
134            *              @ . @ @ @ .
135            *              @ @ . @ . .
136            *              . . . . @ .
137            *              . . . @ @ .
138            *              . . . . . .
139 007F5 8344         NIBHEX 8344C44340     Char 09: sigma
          C443
          40
140            *              . . . . . .
141            *              . . . . . .
```

```
142                *                . @ @ @ @ .
143                *                @ . @ . . .
144                *                @ . . @ . .
145                *                @ . . @ . .
146                *                . @ @ . . .
147                *                . . . . . .
148 007FF F080         NIBHEX F080008782     Char 0A: Linefeed
          0087
          82
149                *                @ . . . . .
150                *                @ . . . . .
151                *                @ . . . . .
152                *                @ @ . @ @ .
153                *                . . . @ . .
154                *                . . . @ @ .
155                *                . . . @ . .
156                *                . . . . . .
157 00809 17A0         NIBHEX 17A0408007     Char 0B: Lambda
          4080
          07
158                *                @ . . . . .
159                *                . @ . . . .
160                *                . . @ . . .
161                *                . @ . @ . .
162                *                @ . . . @ .
163                *                @ . . . @ .
164                *                @ . . . @ .
165                *                . . . . . .
166 00813 CF04         NIBHEX CF0404C304     Char 0C: Mu
          04C3
          04
167                *                . . . . . .
168                *                . . . . . .
169                *                @ . . @ . .
170                *                @ . . @ . .
171                *                @ . . @ . .
172                *                @ . . @ . .
173                *                @ @ @ . @ .
174                *                @ . . . . .
175 0081D 0207         NIBHEX 02078A02F3     Char 0D: Carriage Return
          8A02
          F3
176                *                . . . . @ .
177                *                . . . . @ .
178                *                . . . . @ .
179                *                . . @ . @ .
180                *                . @ . . @ .
181                *                @ @ @ @ @ .
182                *                . @ . . . .
183                *                . . @ . . .
184 00827 8040         NIBHEX 8040C74020     Char 0E: Tau
          C740
          20
185                *                . . . . . .
186                *                . . . . @ .
```

```
187                *                . @ @ @ . .
188                *              @ . @ . . .
189                *              . . @ . . .
190                *              . . @ . . .
191                *              . . @ . . .
192                *              . . . . . .
193 00831 8055          NIBHEX 8055F75580     Char 0F: Phi
          F755
          80
194                *              . @ @ @ . .
195                *              . . @ . . .
196                *              . @ @ @ . .
197                *              @ . @ . @ .
198                *              . @ @ @ . .
199                *              . . @ . . .
200                *              . @ @ @ . .
201                *              . . . . . .
202 0083B E394          NIBHEX E3949494E3     Char 10: Theta
          9494
          E3
203                *              . @ @ @ . .
204                *              @ . . . @ .
205                *              @ . . . @ .
206                *              @ @ @ @ @ .
207                *              @ . . . @ .
208                *              @ . . . @ .
209                *              . @ @ @ . .
210                *              . . . . . .
211 00845 C526          NIBHEX C5262026C5     Char 11: Omega
          2026
          C5
212                *              . . . . . .
213                *              . @ @ @ . .
214                *              @ . . . @ .
215                *              @ . . . @ .
216                *              @ . . . @ .
217                *              . @ . @ . .
218                *              @ @ . @ @ .
219                *              . . . . . .
220 0084F 03A4          NIBHEX 03A4D49403     Char 12: delta
          D494
          03
221                *              . . @ @ . .
222                *              . @ . . . .
223                *              . . @ . . .
224                *              . @ @ @ . .
225                *              @ . . . @ .
226                *              @ . . . @ .
227                *              . @ @ @ . .
228                *              . . . . . .
229 00859 0083          NIBHEX 0083454400     Char 13: epsilon
          4544
          00
230                *              . . . . . .
231                *              . . . . . .
```

```
232             *               . . @ @ . .
233             *               . @ . . . .
234             *               . @ @ . . .
235             *               . @ . . . .
236             *               . . @ @ . .
237             *               . . . . . .
238 00863 40C7          NIBHEX 40C740C740     Char 14: pi
          40C7
          40
239             *               . . . . . .
240             *               . . . . . .
241             *               @ @ @ @ @ .
242             *               . @ . @ . .
243             *               . @ . @ . .
244             *               . @ . @ . .
245             *               . @ . @ . .
246             *               . . . . . .
247 0086D C731          NIBHEX C7312131C7     Char 15: A Umlaut
          2131
          C7
248             *               . @ . @ . .
249             *               . @ @ @ . .
250             *               @ . . . @ .
251             *               @ . . . @ .
252             *               @ @ @ @ @ .
253             *               @ . . . @ .
254             *               @ . . . @ .
255             *               . . . . . .
256 00877 0255          NIBHEX 0255455587     Char 16: a Umlaut
          4555
          87
257             *               . @ . @ . .
258             *               . . . . . .
259             *               . @ @ @ . .
260             *               . . . . @ .
261             *               . @ @ @ @ .
262             *               @ . . . @ .
263             *               . @ @ @ @ .
264             *               . . . . . .
265 00881 C334          NIBHEX C3342434C3     Char 17: O Umlaut
          2434
          C3
266             *               . @ . @ . .
267             *               . @ @ @ . .
268             *               @ . . . @ .
269             *               @ . . . @ .
270             *               @ . . . @ .
271             *               @ . . . @ .
272             *               . @ @ @ . .
273             *               . . . . . .
274 0088B 8354          NIBHEX 8354445483     Char 18: o Umlaut
          4454
          83
275             *               . @ . @ . .
276             *               . . . . . .
```

```
    277             *               . @ @ @ . .
    278             *               @ . . . @ .
    279             *               @ . . . @ .
    280             *               @ . . . @ .
    281             *               . @ @ @ . .
    282             *               . . . . . .
    283 00895 C314          NIBHEX C3140414C3     Char 19: U Umlaut
              0414
              C3
    284             *               . @ . @ . .
    285             *               . . . . . .
    286             *               @ . . . @ .
    287             *               @ . . . @ .
    288             *               @ . . . @ .
    289             *               @ . . . @ .
    290             *               . @ @ @ . .
    291             *               . . . . . .
    292 0089F C314          NIBHEX C31404D304     Char 1A: u Umlaut
              04D3
              04
    293             *               . @ . @ . .
    294             *               . . . . . .
    295             *               @ . . @ . .
    296             *               @ . . @ . .
    297             *               @ . . @ . .
    298             *               @ . . @ . .
    299             *               . @ @ . @ .
    300             *               . . . . . .
    301 008A9 F151          NIBHEX F151170505     Char 1B: Escape
              1705
              05
    302             *               @ @ @ . . .
    303             *               @ . . . . .
    304             *               @ @ . . . .
    305             *               @ . . . . .
    306             *               @ @ @ @ @ .
    307             *               . . @ . . .
    308             *               . . @ @ @ .
    309             *               . . . . . .
    310 008B3 1436          NIBHEX 1436559436     Char 1C: Sigma
              5594
              36
    311             *               @ @ @ @ @ .
    312             *               . @ . . @ .
    313             *               . . @ . . .
    314             *               . . . @ . .
    315             *               . . @ . . .
    316             *               . @ . . @ .
    317             *               @ @ @ @ @ .
    318             *               . . . . . .
    319 008BD 4143          NIBHEX 4143C16141     Char 1D: NotEqual
              C161
              41
    320             *               . . . . . .
    321             *               . . . @ . .
```

```
322                *              @ @ @ @ @ .
323                *              . . @ . . .
324                *              @ @ @ @ @ .
325                *              . @ . . . .
326                *              . . . . . .
327                *              . . . . . .
328 008C7 84E7     9414           NIBHEX 84E7941420     Char 1E: Pound (English)
          20
329                *              . . @ @ . .
330                *              . @ . . @ .
331                *              . @ . . . .
332                *              @ @ @ . . .
333                *              . @ . . . .
334                *              . @ . . . .
335                *              @ @ @ @ . .
336                *              . . . . . .
337 008D1 55A2     55A255A255     NIBHEX 55A255A255     Char 1F:
          55A2
          55
338                *              @ . @ . @ .
339                *              . @ . @ . .
340                *              @ . @ . @ .
341                *              . @ . @ . .
342                *              @ . @ . @ .
343                *              . @ . @ . .
344                *              @ . @ . @ .
345                *              . . . . . .
346 008DB 0000     0000           NIBHEX 0000000000     Char 20: (space)
          0000
          00
347                *              . . . . . .
348                *              . . . . . .
349                *              . . . . . .
350                *              . . . . . .
351                *              . . . . . .
352                *              . . . . . .
353                *              . . . . . .
354                *              . . . . . .
355 008E5 0000     F500           NIBHEX 0000F50000     Char 21: !
          F500
          00
356                *              . . @ . . .
357                *              . . @ . . .
358                *              . . @ . . .
359                *              . . @ . . .
360                *              . . @ . . .
361                *              . . . . . .
362                *              . . . . . .
363                *              . . @ . . .
364 008EF 0070     0070           NIBHEX 0070007000     Char 22: "
          0070
          00
365                *              . @ . @ . .
366                *              . @ . @ . .
```

```
367                *              . @ . @ . .
368                *              . . . . . .
369                *              . . . . . .
370                *              . . . . . .
371                *              . . . . . .
372                *              . . . . . .
373 008F9 41F7     NIBHEX 41F741F741     Char 23: #
          41F7
          41
374                *              . @ . @ . .
375                *              . @ . @ . .
376                *              @ @ @ @ @ .
377                *              . @ . @ . .
378                *              @ @ @ @ @ .
379                *              . @ . @ . .
380                *              . @ . @ . .
381                *              . . . . . .
382 00903 42A2     NIBHEX 42A2F7A221     Char 24: $
          F7A2
          21
383                *              . . @ . . .
384                *              . @ @ @ @ .
385                *              @ . @ . . .
386                *              . @ @ @ . .
387                *              . . @ . @ .
388                *              @ @ @ @ . .
389                *              . . @ . . .
390                *              . . . . . .
391 0090D 3231     NIBHEX 3231804626     Char 25: %
          8046
          26
392                *              @ @ . . . .
393                *              @ @ . . @ .
394                *              . . . @ . .
395                *              . . @ . . .
396                *              . @ . . . .
397                *              @ . . @ @ .
398                *              . . . @ @ .
399                *              . . . . . .
400 00917 6394     NIBHEX 6394650205     Char 26: &
          6502
          05
401                *              . @ . . . .
402                *              @ . @ . . .
403                *              @ . @ . . .
404                *              . @ . . . .
405                *              @ . @ . @ .
406                *              @ . . @ . .
407                *              . @ @ . @ .
408                *              . . . . . .
409 00921 0000     NIBHEX 0000700000     Char 27: '
          7000
          00
410                *              . . @ . . .
411                *              . . @ . . .
```

```
412             *               . . @ . . .
413             *               . . . . . .
414             *               . . . . . .
415             *               . . . . . .
416             *               . . . . . .
417             *               . . . . . .
418 0092B 00C1          NIBHEX 00C1221400      Char 28: (
          2214
          00
419             *               . . . @ . .
420             *               . . @ . . .
421             *               . @ . . . .
422             *               . @ . . . .
423             *               . @ . . . .
424             *               . . @ . . .
425             *               . . . @ . .
426             *               . . . . . .
427 00935 0014          NIBHEX 001422C100      Char 29: )
          22C1
          00
428             *               . @ . . . .
429             *               . . @ . . .
430             *               . . . @ . .
431             *               . . . @ . .
432             *               . . . @ . .
433             *               . . @ . . .
434             *               . @ . . . .
435             *               . . . . . .
436 0093F 80A2          NIBHEX 80A2C1A280      Char 2A: *
          C1A2
          80
437             *               . . . . . .
438             *               . @ . @ . .
439             *               . . @ . . .
440             *               @ @ @ @ @ .
441             *               . . @ . . .
442             *               . @ . @ . .
443             *               . . . . . .
444             *               . . . . . .
445 00949 8080          NIBHEX 8080E38080      Char 2B: +
          E380
          80
446             *               . . . . . .
447             *               . . @ . . .
448             *               . . @ . . .
449             *               @ @ @ @ @ .
450             *               . . @ . . .
451             *               . . @ . . .
452             *               . . . . . .
453             *               . . . . . .
454 00953 000B          NIBHEX 000B070000      Char 2C: ,
          0700
          00
455             *               . . . . . .
456             *               . . . . . .
```

```
    457                 *               . . . . . .
    458                 *               . . . . . .
    459                 *               . @ @ . . .
    460                 *               . @ @ . . .
    461                 *               . . @ . . .
    462                 *               . @ . . . .
    463 00950 80808080  *       NIBHEX 8080808000     Char 2D: -
              8000
    464                 *               . . . . . .
    465                 *               . . . . . .
    466                 *               . . . . . .
    467                 *               @ @ @ @ @ .
    468                 *               . . . . . .
    469                 *               . . . . . .
    470                 *               . . . . . .
    471                 *               . . . . . .
    472 00967 00060600  *       NIBHEX 0006060000     Char 2E: .
              0000
    473                 *               . . . . . .
    474                 *               . . . . . .
    475                 *               . . . . . .
    476                 *               . . . . . .
    477                 *               . . . . . .
    478                 *               . @ @ . . .
    479                 *               . @ @ . . .
    480                 *               . . . . . .
    481 00971 02018040  *       NIBHEX 0201804020     Char 2F: /
              20
    482                 *               . . . . . .
    483                 *               . . . . @ .
    484                 *               . . . @ . .
    485                 *               . . @ . . .
    486                 *               . @ . . . .
    487                 *               @ . . . . .
    488                 *               . . . . . .
    489                 *               . . . . . .
    490 0097B E3159454  *       NIBHEX E3159454E3     Char 30: 0
              E3
    491                 *               . @ @ @ . .
    492                 *               @ . . . @ .
    493                 *               @ . . @ @ .
    494                 *               @ . @ . @ .
    495                 *               @ @ . . @ .
    496                 *               @ . . . @ .
    497                 *               . @ @ @ . .
    498                 *               . . . . . .
    499 00985 0024F704  *       NIBHEX 0024F70400     Char 31: 1
              00
    500                 *               . . @ . . .
    501                 *               . @ @ . . .
```

```
502             *               . . @ . . .
503             *               . . @ . . .
504             *               . . @ . . .
505             *               . . @ . . .
506             *               . @ @ @ . .
507             *               . . . . . .
508 0098F 2615      NIBHEX 2615949464     Char 32: 2
          9494
          64
509             *               . @ @ @ . .
510             *               @ . . . @ .
511             *               . . . . @ .
512             *               . . @ @ . .
513             *               . @ . . . .
514             *               @ . . . . .
515             *               @ @ @ @ @ .
516             *               . . . . . .
517 00999 2294      NIBHEX 2294949463     Char 33: 3
          9494
          63
518             *               . @ @ @ . .
519             *               @ . . . @ .
520             *               . . . . @ .
521             *               . @ @ @ . .
522             *               . . . . @ .
523             *               @ . . . @ .
524             *               . @ @ @ . .
525             *               . . . . . .
526 009A3 8141       NIBHEX 814121F701     Char 34: 4
          21F7
          01
527             *               . . . @ . .
528             *               . . @ @ . .
529             *               . @ . @ . .
530             *               @ . . @ . .
531             *               @ @ @ @ @ .
532             *               . . . @ . .
533             *               . . . @ . .
534             *               . . . . . .
535 009AD 7254      NIBHEX 7254545493     Char 35: 5
          5454
          93
536             *               @ @ @ @ @ .
537             *               @ . . . . .
538             *               @ @ @ @ . .
539             *               . . . . @ .
540             *               . . . . @ .
541             *               @ . . . @ .
542             *               . @ @ @ . .
543             *               . . . . . .
544 009B7 C3A4       NIBHEX C3A4949403     Char 36: 6
          9494
          03
545             *               . . @ @ . .
546             *               . @ . . . .
```

```
547             *               @ . . . . .
548             *               @ @ @ @ . .
549             *               @ . . . @ .
550             *               @ . . . @ .
551             *               . @ @ @ . .
552             *               . . . . . .
553 009C1 1017          NIBHEX 1017905030     Char 37: 7
          9050
          30
554             *               @ @ @ @ @ .
555             *               . . . . @ .
556             *               . . . @ . .
557             *               . . @ . . .
558             *               . @ . . . .
559             *               . @ . . . .
560             *               . @ . . . .
561             *               . . . . . .
562 009CB 6394          NIBHEX 6394949463     Char 38: 8
          9494
          63
563             *               . @ @ @ . .
564             *               @ . . . @ .
565             *               @ . . . @ .
566             *               . @ @ @ . .
567             *               @ . . . @ .
568             *               @ . . . @ .
569             *               . @ @ @ . .
570             *               . . . . . .
571 009D5 6094          NIBHEX 60949492E1     Char 39: 9
          9492
          E1
572             *               . @ @ @ . .
573             *               @ . . . @ .
574             *               @ . . . @ .
575             *               . @ @ @ @ .
576             *               . . . . . .
577             *               . . . @ . .
578             *               . @ @ . . .
579             *               . . . . . .
580 009DF 0063          NIBHEX 0063630000     Char 3A: :
          6300
          00
581             *               . . . . . .
582             *               . @ @ . . .
583             *               . @ @ . . .
584             *               . . . . . .
585             *               . @ @ . . .
586             *               . @ @ . . .
587             *               . . . . . .
588             *               . . . . . .
589 009E9 006B          NIBHEX 006B670000     Char 3B: ;
          6700
          00
590             *               . . . . . .
591             *               . @ @ . . .
```

```
                *              . @ @ . . .
                *              . . . . . .
                *              . @ @ . . .
                *              . @ @ . . .
                *              . . @ . . .
                *              . @ . . . .
009F3 8041      NIBHEX 8041221400     Char 3C: <
      2214
      00
                *              . . . @ . .
                *              . . @ . . .
                *              . @ . . . .
                *              @ . . . . .
                *              . @ . . . .
                *              . . @ . . .
                *              . . . @ . .
                *              . . . . . .
009FD 4141      NIBHEX 4141414141     Char 3D: =
      4141
      41
                *              . . . . . .
                *              . . . . . .
                *              @ @ @ @ @ .
                *              . . . . . .
                *              @ @ @ @ @ .
                *              . . . . . .
                *              . . . . . .
                *              . . . . . .
00A07 1422      NIBHEX 1422418000     Char 3E: >
      4180
      00
                *              @ . . . . .
                *              . @ . . . .
                *              . . @ . . .
                *              . . . @ . .
                *              . . @ . . .
                *              . @ . . . .
                *              @ . . . . .
                *              . . . . . .
00A11 2010      NIBHEX 2010159060     Char 3F: ?
      1590
      60
                *              . @ @ @ . .
                *              @ . . . @ .
                *              . . . . @ .
                *              . . . @ . .
                *              . . @ . . .
                *              . . . . . .
                *              . . @ . . .
                *              . . . . . .
00A1B E314      NIBHEX E314D594E4     Char 40: @
      D594
      E4
                *              . @ @ @ . .
                *              @ .   . @ .
```

```
637                *              @ . @ . @ .
638                *              @ . @ @ @ .
639                *              @ . @ . . .
640                *              @ . . . . .
641                *              . @ @ @ @ .
642                *              . . . . . .
643 00A25 E790          NIBHEX E7909090E7     Char 41: A
          9090
          E7
644                *              . @ @ @ . .
645                *              @ . . . @ .
646                *              @ . . . @ .
647                *              @ @ @ @ @ .
648                *              @ . . . @ .
649                *              @ . . . @ .
650                *              @ . . . @ .
651                *              . . . . . .
652 00A2F F794          NIBHEX F794949463     Char 42: B
          9494
          63
653                *              @ @ @ @ . .
654                *              @ . . . @ .
655                *              @ . . . @ .
656                *              @ @ @ @ . .
657                *              @ . . . @ .
658                *              @ . . . @ .
659                *              @ @ @ @ . .
660                *              . . . . . .
661 00A39 E314          NIBHEX E314141422     Char 43: C
          1414
          22
662                *              . @ @ @ . .
663                *              @ . . . @ .
664                *              @ . . . . .
665                *              @ . . . . .
666                *              @ . . . . .
667                *              @ . . . @ .
668                *              . @ @ @ . .
669                *              . . . . . .
670 00A43 F714          NIBHEX F7141422C1     Char 44: D
          1422
          C1
671                *              @ @ @ @ . .
672                *              @ . . . @ .
673                *              @ . . . @ .
674                *              @ . . . @ .
675                *              @ . . . @ .
676                *              @ . . . @ .
677                *              @ @ @ @ . .
678                *              . . . . . .
679 00A4D F794          NIBHEX F794949414     Char 45: E
          9494
          14
680                *              @ @ @ @ @ .
681                *              @ . . . . .
```

```
682                *              @ . . . . .
683                *              @ @ @ @ . .
684                *              @ . . . . .
685                *              @ . . . . .
686                *              @ @ @ @ @ .
687                *              . . . . . .
688 00A57 F790     9090           NIBHEX F790909010     Char 46: F
          10
689                *              @ @ @ @ @ .
690                *              @ . . . . .
691                *              @ . . . . .
692                *              @ @ @ @ . .
693                *              @ . . . . .
694                *              @ . . . . .
695                *              @ . . . . .
696                *              . . . . . .
697 00A61 E314     1415           NIBHEX E314141527     Char 47: G
          27
698                *              . @ @ @ . .
699                *              @ . . . @ .
700                *              @ . . . . .
701                *              @ . . . . .
702                *              @ . . @ @ .
703                *              @ . . . @ .
704                *              . @ @ @ @ .
705                *              . . . . . .
706 00A6B F780     8080           NIBHEX F7808080F7     Char 48: H
          F7
707                *              @ . . . @ .
708                *              @ . . . @ .
709                *              @ . . . @ .
710                *              @ @ @ @ @ .
711                *              @ . . . @ .
712                *              @ . . . @ .
713                *              @ . . . @ .
714                *              . . . . . .
715 00A75 0014     F714           NIBHEX 0014F71400     Char 49: I
          00
716                *              . @ @ @ . .
717                *              . . @ . . .
718                *              . . @ . . .
719                *              . . @ . . .
720                *              . . @ . . .
721                *              . . @ . . .
722                *              . @ @ @ . .
723                *              . . . . . .
724 00A7F 0304     0404           NIBHEX 03040404F3     Char 4A: J
          F3
725                *              . . . . @ .
726                *              . . . . @ .
```

```
                *              . . . . @ .
                *              . . . . @ .
                *              @ . . . @ .
                *              @ . . . @ .
                *              . @ @ @ . .
                *              . . . . . .
00A89 F780      NIBHEX F780412214     Char 4B: K
      4122
      14
                *              @ . . . @ .
                *              @ . . @ . .
                *              @ . @ . . .
                *              @ @ . . . .
                *              @ . @ . . .
                *              @ . . @ . .
                *              @ . . . @ .
                *              . . . . . .
00A93 F704      NIBHEX F704040404     Char 4C: L
      0404
      04
                *              @ . . . . .
                *              @ . . . . .
                *              @ . . . . .
                *              @ . . . . .
                *              @ . . . . .
                *              @ . . . . .
                *              @ @ @ @ @ .
                *              . . . . . .
00A9D F720      NIBHEX F720C020F7     Char 4D: M
      C020
      F7
                *              @ . . . @ .
                *              @ @ . @ @ .
                *              @ . @ . @ .
                *              @ . @ . @ .
                *              @ . . . @ .
                *              @ . . . @ .
                *              @ . . . @ .
                *              . . . . . .
00AA7 F740      NIBHEX F7408001F7     Char 4E: N
      8001
      F7
                *              @ . . . @ .
                *              @ . . . @ .
                *              @ @ . . @ .
                *              @ . @ . @ .
                *              @ . . @ @ .
                *              @ . . . @ .
                *              @ . . . @ .
                *              . . . . . .
00AB1 E314      NIBHEX E3141414E3     Char 4F: O
      1414
      E3
                *              . @ @ @ . .
                *              @ . . . @ .
```

```
                                *              @ . . . @ .
                                *              @ . . . @ .
                                *              @ . . . @ .
                                *              @ . . . @ .
                                *              . @ @ @ . .
                                *              . . . . . .
00ABB F790                              NIBHEX F790909060     Char 50: P
      9090
      60
                                *              @ @ @ @ . .
                                *              @ . . . @ .
                                *              @ . . . @ .
                                *              @ @ @ @ . .
                                *              @ . . . . .
                                *              @ . . . . .
                                *              @ . . . . .
                                *              . . . . . .
00AC5 E314                              NIBHEX E3141512E5     Char 51: Q
      1512
      E5
                                *              . @ @ @ . .
                                *              @ . . . @ .
                                *              @ . . . @ .
                                *              @ . . . @ .
                                *              @ . @ . @ .
                                *              @ . . @ . .
                                *              . @ @ . @ .
                                *              . . . . . .
00ACF F790                              NIBHEX F790919264     Char 52: R
      9192
      64
                                *              @ @ @ @ . .
                                *              @ . . . @ .
                                *              @ . . . @ .
                                *              @ @ @ @ . .
                                *              @ . @ . . .
                                *              @ . . @ . .
                                *              @ . . . @ .
                                *              . . . . . .
00AD9 6294                              NIBHEX 6294949423     Char 53: S
      9494
      23
                                *              . @ @ @ . .
                                *              @ . . . @ .
                                *              @ . . . . .
                                *              . @ @ @ . .
                                *              . . . . @ .
                                *              @ . . . @ .
                                *              . @ @ @ . .
                                *              . . . . . .
00AE3 1010                              NIBHEX 1010F71010     Char 54: T
      F710
      10
                                *              @ @ @ @ @ .
                                *              . . @ . . .
```

```
817                *                 . . @ . . .
818                *                 . . @ . . .
819                *                 . . @ . . .
820                *                 . . @ . . .
821                *                 . . @ . . .
822                *                 . . . . . .
823 00AED F304          NIBHEX F3040404F3     Char 55: U
            0404
            F3
824                *                 @ . . . @ .
825                *                 @ . . . @ .
826                *                 @ . . . @ .
827                *                 @ . . . @ .
828                *                 @ . . . @ .
829                *                 @ . . . @ .
830                *                 . @ @ @ . .
831                *                 . . . . . .
832 00AF7 7081          NIBHEX 7081068170     Char 56: V
            0681
            70
833                *                 @ . . . @ .
834                *                 @ . . . @ .
835                *                 @ . . . @ .
836                *                 . @ . @ . .
837                *                 . @ . @ . .
838                *                 . . @ . . .
839                *                 . . @ . . .
840                *                 . . . . . .
841 00B01 F702          NIBHEX F7028102F7     Char 57: W
            8102
            F7
842                *                 @ . . . @ .
843                *                 @ . . . @ .
844                *                 @ . . . @ .
845                *                 @ . @ . @ .
846                *                 @ . @ . @ .
847                *                 @ @ . @ @ .
848                *                 @ . . . @ .
849                *                 . . . . . .
850 00B0B 3641          NIBHEX 3641804136     Char 58: X
            8041
            36
851                *                 @ . . . @ .
852                *                 @ . . . @ .
853                *                 . @ . @ . .
854                *                 . . @ . . .
855                *                 . @ . @ . .
856                *                 @ . . . @ .
857                *                 @ . . . @ .
858                *                 . . . . . .
859 00B15 3040          NIBHEX 3040874030     Char 59: Y
            8740
            30
860                *                 @ . . . @ .
861                *                 @ . . . @ .
```

```
862                *              . @ . @ . .
863                *              . . @ . . .
864                *              . . @ . . .
865                *              . . @ . . .
866                *              . . @ . . .
867                *              . . . . . .
868 00B1F 1615          NIBHEX 1615945434     Char 5A: Z
          9454
          34
869                *              @ @ @ @ @ .
870                *              . . . . @ .
871                *              . . . @ . .
872                *              . . @ . . .
873                *              . @ . . . .
874                *              @ . . . . .
875                *              @ @ @ @ @ .
876                *              . . . . . .
877 00B29 00F7          NIBHEX 00F7141400     Char 5B: [
          1414
          00
878                *              . @ @ @ . .
879                *              . @ . . . .
880                *              . @ . . . .
881                *              . @ . . . .
882                *              . @ . . . .
883                *              . @ . . . .
884                *              . @ @ @ . .
885                *              . . . . . .
886 00B33 2040          NIBHEX 2040800102     Char 5C: \
          8001
          02
887                *              . . . . . .
888                *              @ . . . . .
889                *              . @ . . . .
890                *              . . @ . . .
891                *              . . . @ . .
892                *              . . . . @ .
893                *              . . . . . .
894                *              . . . . . .
895 00B3D 0014          NIBHEX 001414F700     Char 5D: ]
          14F7
          00
896                *              . @ @ @ . .
897                *              . . . @ . .
898                *              . . . @ . .
899                *              . . . @ . .
900                *              . . . @ . .
901                *              . . . @ . .
902                *              . @ @ @ . .
903                *              . . . . . .
904 00B47 4020          NIBHEX 4020102040     Char 5E: ^
          1020
          40
905                *              . . @ . . .
906                *              . @ . @ . .
```

```
907                *              @ . . . @ .
908                *              . . . . . .
909                *              . . . . . .
910                *              . . . . . .
911                *              . . . . . .
912                *              . . . . . .
913 00B51 0808     NIBHEX 0808080808     Char 5F: _ (Underline)
          0808
          08
914                *              . . . . . .
915                *              . . . . . .
916                *              . . . . . .
917                *              . . . . . .
918                *              . . . . . .
919                *              . . . . . .
920                *              . . . . . .
921                *              @ @ @ @ @ .
922 00B5B 0030     NIBHEX 0030400000     Char 60: `
          4000
          00
923                *              . @ . . . .
924                *              . @ . . . .
925                *              . . @ . . .
926                *              . . . . . .
927                *              . . . . . .
928                *              . . . . . .
929                *              . . . . . .
930                *              . . . . . .
931 00B65 0245     NIBHEX 0245454587     Char 61: a
          4545
          87
932                *              . . . . . .
933                *              . . . . . .
934                *              . @ @ @ . .
935                *              . . . . @ .
936                *              . @ @ @ @ .
937                *              @ . . . @ .
938                *              . @ @ @ @ .
939                *              . . . . . .
940 00B6F F744     NIBHEX F744444483     Char 62: b
          4444
          83
941                *              @ . . . . .
942                *              @ . . . . .
943                *              @ @ @ @ . .
944                *              @ . . . @ .
945                *              @ . . . @ .
946                *              @ . . . @ .
947                *              @ @ @ @ . .
948                *              . . . . . .
949 00B79 8344     NIBHEX 8344444444     Char 63: c
          4444
          44
950                *              . . . . . .
951                *              . . . . . .
```

```
952             *               . @ @ @ @ .
953             *               @ . . . . .
954             *               @ . . . . .
955             *               @ . . . . .
956             *               . @ @ @ @ .
957             *               . . . . . .
958 00B83 8344          NIBHEX 83444444F7     Char 64: d
          4444
          F7
959             *               . . . . @ .
960             *               . . . . @ .
961             *               . . . . @ .
962             *               . @ @ @ @ .
963             *               @ . . . @ .
964             *               @ . . . @ .
965             *               . @ @ @ @ .
966             *               . . . . . .
967 00B8D 8345          NIBHEX 8345454581     Char 65: e
          4545
          81
968             *               . . . . . .
969             *               . . . . . .
970             *               . @ @ @ . .
971             *               @ . . . @ .
972             *               @ @ @ @ @ .
973             *               @ . . . . .
974             *               . @ @ @ . .
975             *               . . . . . .
976 00B97 80E7          NIBHEX 80E7902000     Char 66: f
          9020
          00
977             *               . . @ . . .
978             *               . @ . @ . .
979             *               . @ . . . .
980             *               @ @ @ . . .
981             *               . @ . . . .
982             *               . @ . . . .
983             *               . @ . . . .
984             *               . . . . . .
985 00BA1 814A          NIBHEX 814A4A4A87     Char 67: g
          4A4A
          87
986             *               . . . . . .
987             *               . . . . . .
988             *               . @ @ @ . .
989             *               @ . . . @ .
990             *               @ . . . @ .
991             *               . @ @ @ @ .
992             *               . . . . @ .
993             *               . @ @ @ . .
994 00BAB F740          NIBHEX F740404087     Char 68: h
          4040
          87
995             *               @ . . . . .
996             *               @ . . . . .
```

```
 997            *              @ @ @ @ . .
 998            *              @ . . . @ .
 999            *              @ . . . @ .
1000            *              @ . . . @ .
1001            *              @ . . . @ .
1002            *              . . . . . .
1003 00BB5 0044         NIBHEX 0044D70400     Char 69: i
           D704
           00
1004            *              . . @ . . .
1005            *              . . . . . .
1006            *              . @ @ . . .
1007            *              . . @ . . .
1008            *              . . @ . . .
1009            *              . . @ . . .
1010            *              . @ @ @ . .
1011            *              . . . . . .
1012 00BBF 0408         NIBHEX 040848D700     Char 6A: j
           48D7
           00
1013            *              . . . @ . .
1014            *              . . . . . .
1015            *              . . @ @ . .
1016            *              . . . @ . .
1017            *              . . . @ . .
1018            *              . . . @ . .
1019            *              @ . . @ . .
1020            *              . @ @ . . .
1021 00BC9 F701         NIBHEX F701824400     Char 6B: k
           8244
           00
1022            *              @ . . . . .
1023            *              @ . . . . .
1024            *              @ . . @ . .
1025            *              @ . @ . . .
1026            *              @ @ . . . .
1027            *              @ . @ . . .
1028            *              @ . . @ . .
1029            *              . . . . . .
1030 00BD3 0014         NIBHEX 0014F70400     Char 6C: l
           F704
           00
1031            *              . @ @ . . .
1032            *              . . @ . . .
1033            *              . . @ . . .
1034            *              . . @ . . .
1035            *              . . @ . . .
1036            *              . . @ . . .
1037            *              . @ @ @ . .
1038            *              . . . . . .
1039 00BDD C740         NIBHEX C740814087     Char 6D: m
           8140
           87
1040            *              . . . . . .
1041            *              . . . . . .
```

```
1042                *               @ @ . @ @ .
1043                *               @ . @ . @ .
1044                *               @ . @ . @ .
1045                *               @ . . . @ .
1046                *               @ . . . @ .
1047                *               . . . . . .
1048 00BE7 C740            NIBHEX C740404087     Char 6E: n
           4040
           87
1049                *               . . . . . .
1050                *               . . . . . .
1051                *               @ @ @ @ . .
1052                *               @ . . . @ .
1053                *               @ . . . @ .
1054                *               @ . . . @ .
1055                *               @ . . . @ .
1056                *               . . . . . .
1057 00BF1 8344            NIBHEX 8344444483     Char 6F: o
           4444
           83
1058                *               . . . . . .
1059                *               . . . . . .
1060                *               . @ @ @ . .
1061                *               @ . . . @ .
1062                *               @ . . . @ .
1063                *               @ . . . @ .
1064                *               . @ @ @ . .
1065                *               . . . . . .
1066 00BFB CF42            NIBHEX CF42424281     Char 70: p
           4242
           81
1067                *               . . . . . .
1068                *               . . . . . .
1069                *               @ @ @ @ . .
1070                *               @ . . . @ .
1071                *               @ . . . @ .
1072                *               @ @ @ @ . .
1073                *               @ . . . . .
1074                *               @ . . . . .
1075 00C05 8142            NIBHEX 81424242CF     Char 71: q
           4242
           CF
1076                *               . . . . . .
1077                *               . . . . . .
1078                *               . @ @ @ @ .
1079                *               @ . . . @ .
1080                *               @ . . . @ .
1081                *               . @ @ @ @ .
1082                *               . . . . @ .
1083                *               . . . . @ .
1084 00C0F C780            NIBHEX C780404040     Char 72: r
           4040
           40
1085                *               . . . . . .
1086                *               . . . . . .
```

```
1087                *              @ . @ @ @ .
1088                *              @ @ . . . .
1089                *              @ . . . . .
1090                *              @ . . . . .
1091                *              @ . . . . .
1092                *              . . . . . .
1093 00C19 8445            NIBHEX 8445454542     Char 73: s
           4545
           42
1094                *              . . . . . .
1095                *              . . . . . .
1096                *              . @ @ @ @ .
1097                *              @ . . . . .
1098                *              . @ @ @ . .
1099                *              . . . . @ .
1100                *              @ @ @ @ . .
1101                *              . . . . . .
1102 00C23 40F3            NIBHEX 40F3440200     Char 74: t
           4402
           00
1103                *              . @ . . . .
1104                *              . @ . . . .
1105                *              @ @ @ . . .
1106                *              . @ . . . .
1107                *              . @ . . . .
1108                *              . @ . @ . .
1109                *              . . @ . . .
1110                *              . . . . . .
1111 00C2D C304            NIBHEX C3040404C7     Char 75: u
           0404
           C7
1112                *              . . . . . .
1113                *              . . . . . .
1114                *              @ . . . @ .
1115                *              @ . . . @ .
1116                *              @ . . . @ .
1117                *              @ . . . @ .
1118                *              . @ @ @ @ .
1119                *              . . . . . .
1120 00C37 C102            NIBHEX C1020402C1     Char 76: v
           0402
           C1
1121                *              . . . . . .
1122                *              . . . . . .
1123                *              @ . . . @ .
1124                *              @ . . . @ .
1125                *              @ . . . @ .
1126                *              . @ . @ . .
1127                *              . . @ . . .
1128                *              . . . . . .
1129 00C41 C304            NIBHEX C3040304C3     Char 77: w
           0304
           C3
1130                *              . . . . . .
1131                *              . . . . . .
```

```
1132                *              @ . . . @ .
1133                *              @ . . . @ .
1134                *              @ . @ . @ .
1135                *              @ . @ . @ .
1136                *              . @ . @ . .
1137                *              . . . . . .
1138 00C4B 4482          NIBHEX 4482018244     Char 78: x
             0182
             44
1139                *              . . . . . .
1140                *              . . . . . .
1141                *              @ . . . @ .
1142                *              . @ . @ . .
1143                *              . . @ . . .
1144                *              . @ . @ . .
1145                *              @ . . . @ .
1146                *              . . . . . .
1147 00C55 C10A          NIBHEX C10A0A0AC7     Char 79: y
             0A0A
             C7
1148                *              . . . . . .
1149                *              . . . . . .
1150                *              @ . . . @ .
1151                *              @ . . . @ .
1152                *              @ . . . @ .
1153                *              . @ @ @ @ .
1154                *              . . . . @ .
1155                *              . @ @ @ . .
1156 00C5F 4446          NIBHEX 444645C444     Char 7A: z
             45C4
             44
1157                *              . . . . . .
1158                *              . . . . . .
1159                *              @ @ @ @ @ .
1160                *              . . . @ . .
1161                *              . . @ . . .
1162                *              . @ . . . .
1163                *              @ @ @ @ @ .
1164                *              . . . . . .
1165 00C69 8063          NIBHEX 8063141400     Char 7B: {
             1414
             00
1166                *              . . @ @ . .
1167                *              . @ . . . .
1168                *              . @ . . . .
1169                *              @ . . . . .
1170                *              . @ . . . .
1171                *              . @ . . . .
1172                *              . . @ @ . .
1173                *              . . . . . .
1174 00C73 0000          NIBHEX 0000F70000     Char 7C: |
             F700
             00
1175                *              . . @ . . .
1176                *              . . @ . . .
```

```
1177                *               . . @ . . .
1178                *               . . @ . . .
1179                *               . . @ . . .
1180                *               . . @ . . .
1181                *               . . @ . . .
1182                *               . . . . . .
1183 00C7D 0014            NIBHEX 0014146380     Char 7D: }
           1463
           80
1184                *               . @ @ . . .
1185                *               . . . @ . .
1186                *               . . . @ . .
1187                *               . . . . @ .
1188                *               . . . @ . .
1189                *               . . . @ . .
1190                *               . @ @ . . .
1191                *               . . . . . .
1192 00C87 8040            NIBHEX 8040800180     Char 7E: ~
           8001
           80
1193                *               . . . . . .
1194                *               . . . . . .
1195                *               . @ . . . .
1196                *               @ . @ . @ .
1197                *               . . . @ . .
1198                *               . . . . . .
1199                *               . . . . . .
1200                *               . . . . . .
1201 00C91 E380            NIBHEX E380808080     Char 7F:
           8080
           80
1202                *               . . . . . .
1203                *               @ . . . . .
1204                *               @ . . . . .
1205                *               @ @ @ @ @ .
1206                *               @ . . . . .
1207                *               @ . . . . .
1208                *               . . . . . .
1209                *               . . . . . .
1210 00C9B                 END
```

```
=ASCII   Abs    1947 #00798 -    57
=INSCUR  Abs    1937 #00791 -    48
=REPCUR  Abs    1927 #00787 -    39
```

Input Parameters

  Source file name is SB&BIT::MS

  Listing file name is SB/BIT:TI:ML::-1

  Object file name is SB%BIT:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
1            *     SSS   BBBB    &     FFFFF   GGG   TTTTT
2            *    S   S  B   B  & &    F      G   G    T
3            *    S      B   B  & &    F      G        T
4            *     SSS   BBBB    &     FFFF   G GGG    T
5            *        S  B   B  & & &  F      G   G    T
6            *    S   S  B   B  &  &   F      G   G    T
7            *     SSS   BBBB    && &  F       GGG     T
8                    TITLE  F & G Shift Table<831212.1206>
9  00C9B             ABS    #00C9B
10           ****************************************************************
11           ****************************************************************
12           **
13           ** Name:(S) FGTBL   -  State table for F & G shifted keys
14           **
15           ** Category:   KEYUTL
16           **
17           ** Purpose:
18           **      This table defines a state machine used to determine
19           **      how to process f and g shifted keys
20           **
21           ** Entry:
22           **      Do not enter
23           **
24           ** Detail:
25           **      The state machine has 7 input bits and 4 output bits.
26           **      The seven input bits are as follows
27           **        Bit 6  F key currently down
28           **        Bit 5  G key currently down
29           **        Bit 4  Some non-FG key newly down
30           **        Bit 3  g annunciator on
31           **        Bit 2  f annunciator on
32           **        Bit 1  Ghost bit
33           **        Bit 0  F or G key was down during last key scan
34           **        The ghost bit is used to indicate that an f or g
35           **        shift has been performed but the annunciator was
36           **        left on because the corresponding key was still
37           **        down.
38           **        The lower 4 bits are stored between key scans
39           **        in the display RAM nibble that contains the f and
40           **        g annunciators.  The lower two bits do not affect
41           **        the display since there are no annunciators in the
42           **        LCD to correspond to these bits.
43           **        These 7 bits form an offset into the table
44           **        which gives the new "state" of the state machine
45           **        and is stored back into display memory.  If
46           **        bit 4 is set but bits 5 and 6 are clear then
47           **        all bits should be cleared following putting the
48           **        f or g modified key codes in the buffer.
49           **
50           ** History:
51           **
52           **    Date     Programmer            Modification
53           **  --------   ----------  ------------------------------------
54           **  10/18/83   B.S.        Updated documentation
55           **
```

```
                *****************************************************************
                *****************************************************************
                *     gf@h gf@h gf@h gf@h gf@h gf@h gf@h gf@h gf@h gf@h gf@h gf@h
                *     0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011
                *FGX
                *000  0000 0000 .... .... 0100 0100 0000 0000 1000 1000 0000 0000
                *       0    0    0    0    4    4    0    0    8    8    0    0
                *001  0000 0000 .... .... 0100 0100 0100 0100 1000 1000 1000 1000
                *       0    0    0    0    4    4    0    0    8    8    0    0
                *010  1001 0001 .... .... 1001 1001 1001 1001 0001 1001 1011 1011
                *       9    1    0    0    9    9    9    9    1    9    B    B
                *011  1011 1011 .... .... 1011 1011 1011 1011 1011 1011 1011 1011
                *       B    B    0    0    B    B    B    B    B    B    B    B
                *100  0101 0001 .... .... 0001 0101 0111 0111 0101 0101 0101 0101
                *       5    1    0    0    1    5    7    7    5    5    5    5
                *101  0111 0111 .... .... 0111 0111 0111 0111 0111 0111 0111 0111
                *       7    7    0    0    7    7    7    7    7    7    7    7
                *110  0000 0000 .... .... 0000 0000 0000 0000 0000 0000 0000 0000
                *       0    0    0    0    0    0    0    0    0    0    0    0
                *111  0000 0000 .... .... 0000 0000 0000 0000 0000 0000 0000 0000
                *       0    0    0    0    0    0    0    0    0    0    0    0

00C9B 0000 =FGTBL  NIBHEX 0000440088000000
      4400
      8800
      0000
00CAB 0000         NIBHEX 0000440088000000
      4400
      8800
      0000
00CBB 9100         NIBHEX 9100999919BB0000
      9999
      19BB
      0000
00CCB BB00         NIBHEX BB00BBBBBBBB0000
      BBBB
      BBBB
      0000
00CDB 5100         NIBHEX 5100157755550000
      1577
      5555
      0000
00CEB 7700         NIBHEX 770077777777
      7777
      7777
00CF7              END
```

```
=FGTBL   Abs    3227 #00C9B -    79
```

Input Parameters

Source file name is SB&FGT::MS

Listing file name is SB/FGT:TI:ML::-1

Object file name is SBXFGT:TI:MS::-1

```
                                111111
                       0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News

```
                *     SSS   BBBB    &     K   K  EEEEE  Y   Y
                *    S   S  B   B  & &    K  K   E      Y   Y
                *    S      B   B  & &    K K    E       Y Y
                *     SSS   BBBB    &     KK     EEEE     Y
                *        S  B   B  & & &  K K    E        Y
                *    S   S  B   B  &  &   K  K   E        Y
                *     SSS   BBBB    && &  K   K  EEEEE    Y
                *
                        TITLE  Keyboard Driver (14 Column Keyboard)<831216.1604>
00CF7                   ABS    #00CF7
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) DEBNCE  -  Debounce and scan keyboard
                ** Name:(S) KEYSCN  -  Scan keyboard
                **
                ** Category:   KEYUTL
                **
                ** Purpose:
                **      Scans keyboard and puts all new keys in key buffer
                **
                ** Entry:
                **
                ** Exit:
                **      P       =  0
                **      DO=(5) =DISINT    (except for WARMST exit)
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: A(W),B(W),C(W),DO
                **
                ** Stk lvls:   0
                **
                ** Detail:
                **      The keyboard is scanned and a bit map of all keys
                **      down is made.  If the number of keys down (not
                **      counting the ON key is greater than 3 then no change
                **      is made to the bit map or key buffer and KEYSCN
                **      returns immediately.  The map is compared to the map
                **      that was made the last time the routine was called.
                **      The new bit map is saved for the next call.  All keys
                **      that have gone down since the last call (up to 7 new
                **      keys) are added to the key buffer (space permitting).
                **      The logical keycodes for unshifted keys that are
                **      generated and stored in the buffer are as follows:
                **
                **   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                **   | Q | W | E | R | T | Y | U | I | O | P | 7 | 8 | 9 | / |
                **   | 01| 02| 03| 04| 05| 06| 07| 08| 09| 0A| 0B| 0C| 0D| 0E|
                **   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                **   | A | S | D | F | G | H | J | K | L | " | 4 | 5 | 6 | * |
                **   | 0F| 10| 11| 12| 13| 14| 15| 16| 17| 18| 19| 1A| 1B| 1C|
                **   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                **   | Z | X | C | V | B | N | M | ( | ) |   | 1 | 2 | 3 | - |
```

```
                **   | 1D| 1E| 1F| 20| 21| 22| 23| 24| 25|eol| 27| 28| 29| 2A|
                **   +---+---+---+---+---+---+---+---+---+ 26+---+---+---+---+
                **   | ON| f |  g|RUN| Lf| Rt|SPC| Up| Dn|   | 0 | . | = | + |
                **   | 2B|   |   | 2E| 2F| 30| 31| 32| 33|   | 35| 36| 37| 38|
                **   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                **
                **      F shifted keys have 56 added to these values.
                **      G shifted keys have 112 added to these values.
                **
                **      The f and g keys themselves are never put in the
                **      buffer.
                **
                **      A state machine is used to control turning on and off
                **      of the f and g annunciators.  See documentation on
                **      FGTBL for further details.
                **
                **      The key buffer looks like this:
                **
                **        1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
                ** +-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+-----------+
                ** | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |           |
                ** +-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+-----------+
                **  ^ ^                                                    ^
                **  | +---- KEYBUF (points to first of 15                  |
                **  |                 bytes of key buffer)                |
                **  +---- KEYPTR (points to nibble that                   |
                **                 tells how many keycodes               |
                **                 buffer contains)                      |
                **             KEYSAV (points to 14 nibbles that         |
                **                     hold previous key bit map) --+
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **   --------  ----------   ---------------------------------
                **   07/16/82  B.S.         Updated documentation
                **   11/16/82  M.B.         Updated exit conditions
                **
                ********************************************************************
                ********************************************************************
                        RDSYMB SBXRAM::MS
                 N512th EQU    7
00CF7 1B8F      =DEBNCE DO=(5) =TIMER1
      3E2
00CFE 1564              C=DATO S
00D02 20                P=     0
00D04 808F              INTOFF
00D08 3200      DBNC10  LC(3)  (N512th)~0
      7
00D0D E6        DBNC20  C=C+1  A
00D0F 800       DBNC30  OUT=CS
00D12 802               A=IN
00D15 A06               C=C+C  P
00D18 5D0               GONC   DBNC35
00D1B 970               ?A=B   W
```

```
109 00D1E D1          GOYES  DBNC40
110 00D20 AF8         B=A    W
111 00D23 54E         GONC   DBNC10        (B.E.T.)
112             *-
113             *-
114 00D26 BF0  DBNC35 ASL    W
115 00D29 BF0         ASL    W
116 00D2C BF0         ASL    W
117 00D2F BF0         ASL    W
118 00D32 5CD         GONC   DBNC30        (B.E.T.)
119             *-
120             *-
121 00D35 20   >3KEYS P=     0
122 00D37 6832        GOTO   KEYS70
123             *-
124             *-
125 00D3B 1524 DBNC40 A=DAT0 S
126 00D3F 942         ?A=C   S
127 00D42 BC          GOYES  DBNC20
128 00D44 AC6         C=A    S
129 00D47 A2E         C=C-1  XS
130 00D4A 52C         GONC   DBNC20
131 00D4D 22  =KEYSCN P=     2
132 00D4F 3F11        LCHEX  0111111111111111
          1111
          1111
          1111
          10
133 00D61 AF1         B=0    W
134 00D64 ACA         A=C    S
135 00D67 B44         A=A+1  S
136 00D6A B44         A=A+1  S             A(S)=3 (Maximum number of keys)
137 00D6D 808F        INTOFF               Next instrs. not interruptable
138 00D71 800 KEYS05  OUT=CS               Activate a key row line
139 00D74 802         A=IN                 Read active key columns
140 00D77 F0          ASL    A
141 00D79 C4          A=A+A  A
142 00D7B 550         GONC   KEYS08        Is ON key down?
143 00D7E B45         B=B+1  S             Yes then turn on bit
144 00D81 C4  KEYS08  A=A+A  A             Shift off left 6 bits
145 00D83 8A8 KEYS10  ?A=0   A             Any bits on in this row?
146 00D86 51          GOYES  KEYS20        No, then go to next row
147 00D88 C4          A=A+A  A             Shift off highest bit
148 00D8A 5B0         GONC   KEYS15        Skip if bit was clear
149 00D8D A01         B=B+C  P             Mark proper bit in map
150 00D90 A4C         A=A-1  S             Decrement number of keys
151 00D93 41A         GOC    >3KEYS        Exit loop if more that 3 keys
152 00D96 0C  KEYS15  P=P+1                Move to next column
153 00D98 5AE         GONC   KEYS10        Loop until all columns done
154 00D9B 22  KEYS20  P=     2             Reset pointer for next row
155 00D9D A76         C=C+C  W             Change constant for next row
156 00DA0 50D         GONC   KEYS05        Loop until all rows done
157 00DA3 20          P=     0
158 00DA5 30F         LCHEX  F
159 00DA8 800         OUT=CS               Leave key rows enabled
```

```
160 00DAB 1B26          DO=(5) =KEYSAV        Point to key map save
          4F2
161 00DB2 15ED          C=DATO 14             Read in last key map
162 00DB6 AE1           B=0    B              Don't shift off non-zero nibs
163 00DB9 BF5           BSR    W
164 00DBC BF5           BSR    W              Shift map to low end of B
165 00DBF AF4           A=B    W              Move new key map to A
166 00DC2 158D          DATO=A 14             Write out new key map
167 00DC6 BFE           C=-C-1 W              1'S complement (NOT) of old keys
168 00DC9 0E76          A=A&C  W              Get all keys that are newly down
169 00DCD 2B            P=     11
170 00DCF 31EE          LCHEX  EE
171 00DD3 0E06          A=A&C  P              Mask out G shift key
172 00DD7 2C            P=     12
173 00DD9 0E06          A=A&C  P              Mask out F shift key
174 00DDD 16A           DO=DO+ (KCOL2)-(KEYSAV)  Point at key column 2 save
175 00DE0 14E           C=DATO B              Read columns containing F and G
176                                             shifts
177 00DE3 AE5           B=C    B              Copy to B(B)
178 00DE6 1907          DO=(2) =DISINT        Point to interrupt disable flag
179 00DEA 1540          DATO=C P              Set interrupt ignore flag
180                                             Note: C(P)<>0
181 00DEE AF2           C=0    W              Needed later (Disable post-clear)
182 00DF1 8080          INTON
183 00DF5 15C0          DATO=C 1              Clear interrupt ignore flag
184                                           (if it isn't already cleared by
185                                            extraneous interrupt caused by
186                                            changing output register with
187                                            a key down and interrupts
188                                            disabled)
189 00DF9 20            P=     0
190             *    B(B) = xxxF xxxG
191 00DFB 3111          LCHEX  11
192 00DFF 0E61          B=B&C  B              Mask out all but F & G shifts
193 00E03 A05           B=B+B  P
194 00E06 A05           B=B+B  P
195 00E09 A05           B=B+B  P              Shift G
196             *    B(B) = 000F G000
197 00E0C A65           B=B+B  B
198 00E0F A65           B=B+B  B              Shift F and G
199             *    B(B) = 0FG0 0000
200 00E12 96D           ?B#0   B              Are either F or G shift down?
201 00E15 50            GOYES  FG10           Yes, then skip
202 00E17 A4E           C=C-1  S              No, then enable post-clear
203 00E1A 2D    FG10    P=     13
204 00E1C 918           ?A=0   WP             Any non-shift keys newly down?
205 00E1F 80            GOYES  FG15           No, then disable post-clear
206 00E21 A61           B=B+C  B              Yes, then increment B(1)
207 00E24 550           GONC   FG20           (B.E.T.)  Skip
208 00E27 AC2   FG15    C=0    S              Disable post-clear
209 00E2A 20    FG20    P=     0
210             *    B(B) = 0FGX 000?
211 00E2C 31B5          LCHEX  5B             Limit of valid table
212 00E30 AED           BCEX   B              B(B)=limit, C(A)=index
213 00E33 1B20          DO=(5) =ANNAD2        Point to shift annunciators
```

```
          1E2
214 00E3A 1560        C=DATO P          Read annunciator nibble
215             *   C(B) = OFGX gf@h
216 00E3E 9E1         ?B>C   B          Is index in table?
217 00E41 80          GOYES  FG30       Yes, then read table
218 00E43 A82         C=0    P          No, then value is zero
219 00E46 5B1         GONC   FG40       (B.E.T.)
220 00E49 D5   FG30    B=C    A          Copy index to B(A)
221 00E4B 3400        LC(5)  =FGTBL     Start of table
          000
222 00E52 C9          C=C+B  A
223 00E54 134         DO=C              Point into table
224 00E57 1560        C=DATO P          Read table entry
225 00E5B 1B20        DO=(5) =ANNAD2    Point back at annunciators
          1E2
226 00E62 1540 FG40   DATO=C P          Write out new annunciators
227 00E66 AA2         C=0    XS
228 00E69 A06         C=C+C  P          Is G shift on?
229 00E6C 4B0         GOC    FG50       Yes, then increment C(S) twice
230 00E6F A06         C=C+C  P          Is F shift on?
231 00E72 480         GOC    FG51       Yes, then increment C(S) once
232 00E75 580         GONC   FG52       (B.E.T.) No, then skip
233             *-
234             *-
235 00E78 B26  FG50   C=C+1  XS
236 00E7B B26  FG51   C=C+1  XS
237 00E7E AE2  FG52   C=0    B
238 00E81 AF1         B=0    W
239 00E84 2D          P=     13
240             *
241 00E86 918  KEYS25 ?A=0   WP         Are any new keys here?
242 00E89 D2          GOYES  KEYS40     No, then finish up
243 00E8B 90C  KEYS30 ?A#0   P          Are any keys down in this nibble?
244 00E8E 31          GOYES  KEYS35     Yes, then find them
245 00E90 0D          P=P-1             No, then skip these 4 bits
246 00E92 B66         C=C+1  B
247 00E95 B66         C=C+1  B
248 00E98 B66         C=C+1  B
249 00E9B B66         C=C+1  B          Skip 4 keycodes
250 00E9E 5CE         GONC   KEYS30     (B.E.T.) Loop back for next nib
251             *-
252             *-
253 00EA1 B66  KEYS35 C=C+1  B          Add one to keycode
254 00EA4 A14         A=A+A  WP         Shift, was high bit set?
255 00EA7 59F         GONC   KEYS35     No, then loop to next bit
256 00EAA BF1         BSL    W
257 00EAD BF1         BSL    W          Shift keycode list
258 00EB0 AE5         B=C    B          Add this keycode to list
259 00EB3 42D         GOC    KEYS25     (B.E.T.) Loop back for more keys
260 00EB6 94A  KEYS40 ?C=0   S          Is post-clear required?
261 00EB9 90          GOYES  KEYS45     No, then skip
262 00EBB AC2         C=0    S
263 00EBE 1544        DATO=C S          Clear annunciators
264 00EC2 80D2 KEYS45 P=C    2
265 00EC6 80FF        CPEX   15         P=0,C(S)=C(XS)
```

```
266 00ECA 969   KEYS46 ?B=0   B            Is the keycode list empty?
267 00ECD 74           GOYES  KEYS6j       Yes, then check for special ON
268                                           key processing
269 00ECF A6D          B=B-1  B            Convert to option base 0
270 00ED2 3130         LCHEX  03           Two bit mask
271 00ED6 AEA          A=C    B            Copy mask to A(B)
272 00ED9 BEC          A=-A-1 B            Complement mask
273 00EDC 0E60         A=A&B  B            Don't shift off any non zero bits
274 00EE0 81C          ASRB
275 00EE3 81C          ASRB                Divide by 4
276 00EE6 0E65         C=C&B  B            Get mod 4 of keycode
277 00EEA A66          C=C+C  B            x2
278 00EED AE5          B=C    B            =2x
279 00EF0 A66          C=C+C  B            x4
280 00EF3 A61          B=B+C  B            =6x
281 00EF6 A66          C=C+C  B            x8
282 00EF9 A61          B=B+C  B            =14x
283 00EFC A68          B=B+A  B            Row major keycode
284 00EFF 3183         LC(2)  56
285 00F03 80DF         P=C    15           Recall FG shift status
286 00F07 890   KEYS48 ?P=    0            Shifted?
287 00F0A D0           GOYES  KEYS49       No, then continue
288 00F0C 0D           P=P-1               Decrement shift status
289 00F0E A61          B=B+C  B            Add back in shift
290 00F11 55F          GONC   KEYS48       (B.E.T.)  Loop back
291             *-
292             *-
293 00F14 474   KEYS6j GOC    KEYS60       (B.E.T.)
294             *-
295             *-
296 00F17 B65   KEYS49 B=B+1  B            Convert back to option base 1
297 00F1A 3424         LC(5)  (=KEYBUF)-2  Pointer to key buffer-2
          4F2
298 00F21 1A34         DO=(4) =KEYPTR      Point to buffer pointer
          4F
299 00F27 D0           A=0    A
300 00F29 808F         INTOFF
301 00F2D 1520         A=DATO P            Read in one buffer pointer
302 00F31 B04          A=A+1  P            Increment buffer pointer
303 00F34 431          GOC    KEYS50       Is buffer full? Then exit
304 00F37 1500         DATO=A P            Write out new buffer pointer
305 00F3B C2           C=C+A  A
306 00F3D C2           C=C+A  A            Point to proper buffer loc
307 00F3F 134          DO=C                Move pointer to DO
308 00F42 AE9          C=B    B            Move keycode to C
309 00F45 14C          DATO=C B            Write out keycode to buffer
310 00F48 8080  KEYS50 INTON               Re-enable interrupts
311 00F4C AE1          B=0    B            Don't shift off non-zero nibs
312 00F4F BF5          BSR    W
313 00F52 BF5          BSR    W
314 00F55 460          GOC    KEYS60       Exit if buffer overflowed
315 00F58 617F         GOTO   KEYS46       Loop back for next keycode
316             * S P E C I A L   O N   K E Y   P R O C E S S I N G
317 00F5C 301   KEYS60 LCHEX  1
318 00F5F 808F         INTOFF
```

```
319 00F63 800          OUT=CS              Activate ON key row
320 00F66 803          C=IN
321 00F69 F2           CSL    A
322 00F6B C6           C=C+C  A            Is ON key down?
323 00F6D 4F1          GOC    KEYS80       Yes, then do special processing
324 00F70 30F  KEYS70  LCHEX  F
325 00F73 800          OUT=CS              Activate all key rows before
326                                          returning
327 00F76 1B07         DO=(5) =DISINT
          4F2
328 00F7D 15C0         DATO=C 1            Disable interrupt that may occur
329 00F81 8080         INTON                  following this instruction
330 00F85 D2           C=0    A
331 00F87 15C0         DATO=C 1            Reenable interrupts
332 00F8B 03           RTNCC               Return from KEYSCN
333             *-
334             *-
335 00F8D       KEYS80
336 00F8D 308          LCHEX 8             Activate row containing "/" key
337 00F90 800          OUT=CS
338 00F93 803          C=IN
339 00F96 F2           CSL    A
340 00F98 C6           C=C+C  A
341 00F9A C6           C=C+C  A
342 00F9C C6           C=C+C  A            Is "/" key down?
343 00F9E 445          GOC    WARMST       Yes, then perform warm start
344 00FA1 301          LCHEX  1
345 00FA4 800          OUT=CS              Activate key row containing ON
346 00FA7 803          C=IN
347 00FAA F2           CSL    A
348 00FAC C6           C=C+C  A            Is ON key down?
349 00FAE 4ED          GOC    KEYS80       Yes, loop back to continue scan
350 00FB1 94E          ?C#0   S            Is the ON key f or g shifted?
351 00FB4 CB           GOYES  KEYS70       Yes, then exit KEYSCN
352 00FB6 1A14         DO=(4) =ATNDIS
          4F
353 00FBC 1560         C=DATO P            Read attn disable flag
354 00FC0 90E          ?C#0   P            Is it set?
355 00FC3 DA           GOYES  KEYS70       Yes, then exit KEYSCN
356 00FC5 160          DO=DO+ (ATNFLG)-(ATNDIS)
357 00FC8 3301         LCHEX  2B10         Set key buf= ON key,
          B2
358                                          (leave room for ATNFLG
359 00FCE 1560         C=DATO P            Read ATNFLG
360 00FD2 144          DATO=C A            Initialize key buffer
361 00FD5 A0E          C=C-1  P            Decrement ATNFLG
362 00FD8 90A          ?C=0   P            Would this wrap it around to zero
363 00FDB 50           GOYES  KEYS85       Yes, then leave ATNFLG at 1
364 00FDD 14C          DATO=C B            Re-write out decremented ATNFLG
365 00FE0 850  KEYS85  ST=1   =Except      Set exception flag
366 00FE3 D2           C=0    A
367 00FE5 1A20         DO=(4) =ANNAD2
          1E
368 00FEB 15C0         DATO=C 1            Clear f and g annunciator status
369 00FEF 608F         GOTO   KEYS70
```

```
370             *-
```

```
                        EJECT
                *****************************************************************
                *****************************************************************
                **
                ** Name:    WARMST  -  Warm Start
                **
                ** Category:   KEYUTL
                **
                ** Purpose:
                **      Prompts user for recovery level.  Allows control to
                **      be regained from microcode infinite loops.
                **
                ** Entry:
                **
                ** Exit:
                **      Depends on user response.  See Detail.
                **
                ** Detail:
                **      "INIT:" is sent to display.  The display is
                **      turned on and the keyboard is scanned.  If the user
                **      presses a digit key in the valid range (below) the
                **      digit is displayed.  When the End Line key is pushed
                **      the warm start requested is performed.  The warm
                **      start levels are as follows:
                **
                **          1: Clear USER mode then jump to MAINLP
                **          2: Perform ROM test, Clear USER then jump to MAINLP
                **          3: Cold Start
                **
                ** History:
                **
                **    Date      Programmer          Modification
                **   --------   ----------   ---------------------------------
                **   07/22/82   B.S.         Rewrote in present form to use
                **                           display routines.
                **   01/23/83   B.S.         Added check for CALC mode before
                **                           jumping to MAINLP
                **
                *****************************************************************
                *****************************************************************
00FF3 31E8  =WARMST LCHEX  8E               Beep parameters
00FF7 8E00          GOSUBL =RCKBp           Make a beep
      00
00FFD 7320          GOSUB  CLRXDS           Disable external display
01001 803   WARMSO  C=IN
01004 8AE           ?C#0   A                Are all keys up?
01007 AF            GOYES  WARMSO           No, then continue to wait
01009 8E00          GOSUBL =nokeys          Clear key buffer
      00
0100F 8E00          GOSUBL =LCDINI          Initialize LCD
      00
                                              (Row Drivers, Contrast on)
01015 3463          LC(5)  WARMS+
      010
0101C 8080          INTON                   Need interrupts on
```

```
422 01020 06          RSTK=C
423 01022 0F          RTI                    Reenable interrupts
424           *-
425           *-
426 01024 D2  =CLRXDS C=0    A
427 01026 1B47        DO=(5) =DSPCHX
          6F2
428 0102D 144         DAT0=C A               Disable external display devices
429 01030 01          RTN
430           *-
431           *-
432 01032 0C   WARM3D P=P+1
433 01034 0C   WARM2D P=P+1
434 01036     WARMS+
435 01036 D2   WARM1D C=0    A
436 01038 80F0        CPEX   0
437 0103C 108         R0=C
438 0103F 8E00        GOSUBL =DSPRST
          00
439 01045 1FCD        D1=(5) =INITMS
          010
440 0104C 7000        GOSUB  =Bf2Dsp
441 01050 110         A=R0
442 01053 3113        LC(2)  \1\
443 01057 C2          C=A+C  A
444 01059 8E00        GOSUBL =DSPCHC
          00
445 0105F 8E00        GOSUBL =BLDDSP
          00
446 01065 7580  WARM2O GOSUB  POPBUF
447 01069 4BF         GOC    WARM2O
448 0106C D4          A=B    A
449 0106E 8E00        GOSUBL =FINDA
          00
450 01074 00          CON(2) =k#1            Warm Start 1:USER mode off
451 01076 0CF         REL(3) WARM1D
452 01079 00          CON(2) =k#2            Warm Start 2:ROM test
453 0107B 9BF         REL(3) WARM2D
454 0107E 00          CON(2) =k#3            Warm Start 3:Clear all
455 01080 2BF         REL(3) WARM3D
456 01083 00          CON(2) =k#EOL          Endline:  Do it.
457 01085 800         REL(3) WARMEX
458 01088 00          CON(2) 0
459 0108A 5AD         GONC   WARM2O          (B.E.T.)
460           *-
461           *-
462 0108D 118  WARMEX C=R0
463 01090 CE          C=C-1  A
464 01092 431         GOC    WARM1X
465 01095 CE          C=C-1  A
466 01097 480         GOC    WARM2X
467 0109A 8C00  WARM3X GOLONG =CLDST1
          00
468           *-
469           *-
```

```
470 010A0 8E00  WARM2X GOSUBL =ROMTST        Test ROM and go to MAINLP
          00
471 010A6 3100 =WARM1X LC(2)  =flUSER
472 010AA 8E00         GOSUBL =sflagc        Clear USER mode
          00
473 010B0 8E00         GOSUBL =CRLFND
          00
474 010B6 8E00         GOSUBL =fCALC?
          00
475 010BC 580          GONC   MAINj
476 010BF 8C00         GOLONG =CALCj
          00
477             *-
478             *-
479 010C5      =CONFMN                       Configure then jump to main loop
480 010C5 840   MAINj  ST=0   0
481 010C8 8F00         GOSBVL =NOSCRL
          000
482 010CF 8F00         GOSBVL =CONF
          000
483 010D6 8C00         GOLONG =MAINLP
          00
484             *-
485             *-
486 010DC B1   =INITMS NIBHEX B1             Cursor off, INIT:
487 010DE C394         NIBASC \<INIT: \
          E494
          45A3
          02
488 010EC FF           NIBHEX FF
489             *-
```

```
490                     EJECT
491                 ********************************************************************
492                 ********************************************************************
493                 **
494                 ** Name:(S) POPBUF  -  Pop Key Buffer
495                 **
496                 ** Category:   KEYUTL
497                 **
498                 ** Purpose:
499                 **      Pops a key from keyboard buffer into B(A)
500                 **
501                 ** Entry:
502                 **
503                 ** Exit:
504                 **      Carry set    ==> Key buffer was empty
505                 **           clear  ==> B(A) contains keycode
506                 **      Key# just popped has been copied to last position
507                 **        in keybuffer.
508                 **
509                 ** Calls:      None
510                 **
511                 ** Uses.......
512                 **  Inclusive: C(W),B(A),D0
513                 **
514                 ** Stk lvls:   0
515                 **
516                 ** Detail:
517                 **      Disables interrupts and pops a key from buffer.
518                 **
519                 ** History:
520                 **
521                 **    Date      Programmer              Modification
522                 **  --------   ----------   ------------------------------------
523                 **  07/16/82   B.S.         Updated documentation
524                 **  11/04/82   NM           Add copy of last key to key14 slot
525                 **
526                 ********************************************************************
527                 ********************************************************************
528 010EE 1B34 =POPBUF D0=(5) =KEYPTR
            4F2
529 010F5 1562          C=DAT0 XS               Buffer entry count
530 010F9 A2E           C=C-1  XS               Empty?
531 010FC 400           RTNC                    Yes.
532 010FF 808F          INTOFF                  No.
533 01103 1542          DAT0=C XS               Update entry count.
534 01107 160           D0=D0+ 1                Point at keybuffer
535 0110A 1567          C=DAT0 W
536 0110E D1            B=0    A
537 01110 AE5           B=C    B                Pop top key.
538 01113 816           CSRC
539 01116 816           CSRC                    Shift off top key
540 01119 15CD          DAT0=C 14               Update first 7 chars
541 0111D 16F           D0=D0+ 16
542 01120 15ED          C=DAT0 14               Read top 7 chars
543 01124 181           D0=D0- 2
```

```
544 01127 1547          DATO=C W              Update rest of buffer w/last key
545 0112B 8080          INTON
546 0112F 03            RTNCC
547 01131               END
```

```
 >3KEYS  Abs    3381 #00D35 -   121   151
 ANNAD2  Abs  188674 #2E102 -    96   213   225   367
 ATNDIS  Abs  193601 #2F441 -    96   352   356
 ATNFLG  Abs  193602 #2F442 -    96   356
 BLDDSP  Ext                -   445
 Bf2Dsp  Ext                -   440
 CALCj   Ext                -   476
 CLDST1  Ext                -   467
=CLRXDS  Abs    4132 #01024 -   426   413
 CONF    Ext                -   482
=CONFMN  Abs    4293 #010C5 -   479
 CRLFND  Ext                -   473
 DBNC10  Abs    3336 #00D08 -   102   111
 DBNC20  Abs    3341 #00D0D -   103   127   130
 DBNC30  Abs    3343 #00D0F -   104   118
 DBNC35  Abs    3366 #00D26 -   114   107
 DBNC40  Abs    3387 #00D3B -   125   109
=DEBNCE  Abs    3319 #00CF7 -    98
 DISINT  Abs  193648 #2F470 -    96   178   327
 DSPCHC  Ext                -   444
 DSPCHX  Abs  194164 #2F674 -    96   427
 DSPRST  Ext                -   438
 Except  Ext                -   365
 FG10    Abs    3610 #00E1A -   203   201
 FG15    Abs    3623 #00E27 -   208   205
 FG20    Abs    3626 #00E2A -   209   207
 FG30    Abs    3657 #00E49 -   220   217
 FG40    Abs    3682 #00E62 -   226   219
 FG50    Abs    3704 #00E78 -   235   229
 FG51    Abs    3707 #00E7B -   236   231
 FG52    Abs    3710 #00E7E -   237   232
 FGTBL   Ext                -   221
 FINDA   Ext                -   449
=INITMS  Abs    4316 #010DC -   486   439
 KCOL2   Abs  193645 #2F46D -    96   174
 KEYBUF  Abs  193604 #2F444 -    96   297
 KEYPTR  Abs  193603 #2F443 -    96   298   528
 KEYS05  Abs    3441 #00D71 -   138   156
 KEYS08  Abs    3457 #00D81 -   144   142
 KEYS10  Abs    3459 #00D83 -   145   153
 KEYS15  Abs    3478 #00D96 -   152   148
 KEYS20  Abs    3483 #00D9B -   154   146
 KEYS25  Abs    3718 #00E86 -   241   259
 KEYS30  Abs    3723 #00E8B -   243   250
 KEYS35  Abs    3745 #00EA1 -   253   244   255
 KEYS40  Abs    3766 #00EB6 -   260   242
 KEYS45  Abs    3778 #00EC2 -   264   261
 KEYS46  Abs    3786 #00ECA -   266   315
 KEYS48  Abs    3847 #00F07 -   286   290
 KEYS49  Abs    3863 #00F17 -   296   287
 KEYS50  Abs    3912 #00F48 -   310   303
 KEYS60  Abs    3932 #00F5C -   317   293   314
 KEYS6j  Abs    3860 #00F14 -   293   267
 KEYS70  Abs    3952 #00F70 -   324   122   351   355   369
 KEYS80  Abs    3981 #00F8D -   335   323   349
```

```
 KEYS85  Abs    4064 #00FE0 -   365   363
 KEYSAV  Abs  193634 #2F462 -    96   160   174
=KEYSCN  Abs    3405 #00D4D -   131
 LCDINI  Ext         #     -   418
 MAINLP  Ext         #     -   483
 MAINj   Abs    4293 #010C5 -   480   475
 N512th  Abs       7 #00007 -    97   102
 NOSCRL  Ext         #     -   481
=POPBUF  Abs    4334 #010EE -   528   446
 RCKBp   Ext         #     -   412
 ROMTST  Ext         #     -   470
 TIMER1  Abs  189432 #2E3F8 -    96    98
 WARM1D  Abs    4150 #01036 -   435   451
=WARM1X  Abs    4262 #010A6 -   471   464
 WARM2D  Abs    4197 #01065 -   446   447   459
 WARM2D  Abs    4148 #01034 -   433   453
 WARM2X  Abs    4256 #010A0 -   470   466
 WARM3D  Abs    4146 #01032 -   432   455
 WARM3X  Abs    4250 #0109A -   467
 WARMEX  Abs    4237 #0108D -   462   457
 WARMS+  Abs    4150 #01036 -   434   420
 WARMSO  Abs    4097 #01001 -   414   416
=WARMST  Abs    4083 #00FF3 -   411   343
 fCALC?  Ext         #     -   474
 flUSER  Ext         #     -   471
 k#1     Ext         #     -   450
 k#2     Ext         #     -   452
 k#3     Ext         #     -   454
 k#EOL   Ext         #     -   456
 nokeys  Ext         #     -   417
 sflagc  Ext         #     -   472
```

Input Parameters

  Source file name is SB&KEY::MS

  Listing file name is SB/KEY:TI:ML::-1

  Object file name is SBXKEY:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            DEFFIL  STITLE Fetch Default File Spec
 2
 3            *           TTTTT  IIIII   &     U   U  TTTTT  L
 4            *             T      I    & &    U   U    T    L
 5            *             T      I    & &    U   U    T    L
 6            *             T      I     &     U   U    T    L
 7            *             T      I    & & &  U   U    T    L
 8            *             T      I    &  &   U   U    T    L
 9            *             T    IIIII   && &   UUU     T    LLLLL
10
11                    TITLE  Command Execution Utilities   <831212.1208>
12 01131              ABS    #1131
13                    RDSYMB TI%EQU::MS
14                    RDSYMB SB%RAM::MS
15
16           ****************************************************************
17           ****************************************************************
18           **
19           ** Name:    DEFFIL  -  Fetch Default (Current) File Spec
20           **
21           ** Category:   FILUTL
22           **
23           ** Purpose:
24           **      Fetches current file name and device code into
25           **      the same registers as FSPECx.
26           **
27           ** Entry:
28           **      P       =  0
29           **
30           ** Exit:
31           **      A       =  File name
32           **      R0(3-0) =  Two blanks (characters 9, 10 of file name)
33           **      D(S)    =  Device type as returned from FSPECx
34           **                 (MAIN = 0, IRAM = 1, ROM = 2, etc)
35           **      D(A)    =  0 if MAIN
36           **                 =  D(B) = Port ID if PORT, rest 0
37           **      P       =  0
38           **      Carry   =  Clear
39           **
40           ** Calls:      CURDVC, D1=CRS, FLDEVX
41           **
42           ** Uses.......
43           **  Exclusive: A, C, D(A), R0, D1
44           **  Inclusive: A, B, C, D, R0, R1, R2, D1, S2
45           **
46           ** Stk lvls:   4
47           **
48           ** History:
49           **
50           **    Date     Programmer              Modification
51           **  --------   ----------   ---------------------------------
52           **  06/08/82      FH        Designed and coded.
53           **
54           ****************************************************************
55           ****************************************************************
```

```
     56 01131 8F00 =DEFFIL GOSBVL =CURDVC        Find current device
              000
     57 01138 7810         GOSUB  FLDEVX         Clean up D(A)
     58 0113C 8E00         GOSUBL =D1=CRS        Read current file name
              00
     59 01142 1537         A=DAT1 W               .
     60 01146 3302         LCASC  \  \           Load two trailing blanks
              02
     61 0114C 108          RO=C                   .
     62 0114F 03           RTNCC
```

```
FLDEVX  STITLE Make File Device Code Explicit
*****************************************************************
*****************************************************************
**
** Name:(S) FLDEVX  -  Make Device Code Explicit
** Name:    FLDEV+  -  Make Device Code Explicit
**
** Category:   EXCUTL
**
** Purpose:
**      Maps the FSPECx device code into the FIB device code
**      without having to find the file using FINDF.  In certain
**      cases maps unspecified device states to appropriate code.
**
**    For SOURCE device:
**      Does NOT map undefined device to MAIN.  Identifies port
**      if explicit.  Returns carry set only for illegal port.
**
**    For DESTINATION device:
**      Maps undefined device to MAIN, explicitly identifies
**      port.  Returns carry set for illegal or unspecified
**      port.
**
** Entry:
**      S3(sDEST) = 0 if SOURCE file (see above), 1 if DEST file.
**      P      =  0
**    FLDEVX:
**      D(S)   =  Device code returned from FSPECx.
**      D(3-0) =  Device code data returned from FSPECx.
**    FLDEV+:  (for file info as returned by RDINFO)
**      D(0)   =  Device code returned from FSPECx.
**      D(4-1) =  Device code data returned from FSPECx.
**
** Exit:
**      P      =  0
**    Carry clear:
**      Device code and data are sufficiently explicit.
**      D(S)   =  See Detail
**      D(A)   =  See Detail
**    Carry set:
**      Device code and data are illegal or not explicit:
**        SOURCE:  Port ID is specified but illegal.
**                 D(A) = 0
**        DEST:    Port ID unspecified or illegal.
**                 If PORT ID unspecified:  D(B) = FF
**                    else                  D(A) = 0
**
**      C(3-0) =  Error code: "Device not Found"
**
** Calls:      ROMF-1, CSLW5, CSRW5
**
** Uses.......
**  Exclusive: C(S), C(A), D(A), R0(15-5),
**  Inclusive: B, C, D, D1,      R0(15-5), R1, R2, R3, S2
**
```

```
118                ** Stk lvls:   3
119                **
120                ** Detail:
121                **          ON ENTRY                    ON EXIT
122                **       ------------   ----------------------------------------
123                **          D(S)           D(S)     D(4-3)   D(XS)      D(B)
124                **       ------------   ---------- ------  -------  ----------
125                **        F  (Undef)    0 (DEST)       0        0         0
126                **                      F (SOURCE)     0        0         0
127                **        0  (MAIN)     0              0        0         0
128                **        1  (PORT)     1 (IRAM)       0        0       Port ID
129                **                      2 (ROM)        0        0       Port ID
130                **                      3 (EEPROM)     0        0       Port ID
131                **        7  (CARD)     7 (CARD)       entry    entry   PCRD flg
132                **        8+ (HPIL+)    8+             entry    <device address>
133                **
134                **
135                ** History:
136                **
137                **     Date      Programmer          Modification
138                **    --------   ----------   ---------------------------------
139                **    05/19/82      FH        Wrote.
140                **    11/15/82      FH        Completely rewrote for new device
141                **                             codes.
142                **
143                **  03/21/83        JP        Error Msg = eDVCNF
144                **  03/21/83        JP        Pack byte by calling ROMF-
145                **  03/21/83        JP        If PORT not found, set D(A)=0
146                **
147                ********************************************************************
148                ********************************************************************
149 01151 817  =FLDEV+ DSRC                      Shift RDINFO file data
150 01154 D2   =FLDEVX C=0     A
151 01156 ACB          C=D     S                 Fetch device code
152 01159 B46          C=C+1   S                 Device defined?
153 0115C 531          GONC    FLDE60            GOYES
154 0115F D3           D=0     A                 Clear D(A)
155 01161 863          ?ST=0   sDEST             Don't map to MAIN?
156 01164 50           GOYES   FLDE20
157 01166 AC3          D=0     S                 Map to MAIN
158 01169 ABB   FLDE20 C=D     X                 Fetch device addr if present
159 0116C D7    FLDE40 D=C     A                 Install cleaned up code
160 0116E 03    FLDE50 RTNCC
161 01170 A46   FLDE60 C=C+C   S                 CARD or HPIL+? ( 7 <= D(S) < F )
162 01173 4AF          GOC     FLDE50            If so, return D(A) as is
163 01176 948          ?D=0    S                 MAIN?
164 01179 3F           GOYES   FLDE40            Clear D(A), RTNCC
165 0117B AEB          C=D     B                 Fetch port ID
166 0117E D7           D=C     A                  . and restore cleaned up data
167 01180 B66          C=C+1   B                 Port specified?
168 01183 590          GONC    FLDE80
169 01186 873          ?ST=1   sDEST             Error?
170 01189 D3           GOYES   FLDE90
171 0118B 03           RTNCC
172 0118D 102   FLDE80 R2=A
```

```
173 01190 118          C=R0                    Save D0 in R0
174 01193 8F00         GOSBVL =CSLW5            . and preserve file name
          000
175 0119A 136          CD0EX                    .
176 0119D 108          R0=C                     .
177 011A0 8F00         GOSBVL =ROMF-           Identify port
          000
178               *
179               * If PORT not found (Carry Set)
180               *    Set D(A)=0 to distinquish from PORT not specified D(B)=FF
181               *
182 011A7 540          GONC   FLDE85
183 011AA D3           D=0    A
184 011AC 112   FLDE85 A=R2                    Recall file name
185 011AF 118          C=R0                    Restore D0
186 011B2 134          D0=C                     .
187 011B5 8F00         GOSBVL =CSRW5           Restore last chars of file name
          000
188 011BC 108          R0=C                     .
189 011BF D2           C=0    A                Clean up D(A)
190 011C1 AEB          C=D    B                 .
191 011C4 D7           D=C    A                 .
192               *
193 011C6 3300 =FLDE90 LC(4)  =eDVCNF          Load error code (MUST BE LC(4))
          00
194 011CC 01           RTN                     Return carry set or clear
```

```
195             FILFIL  STITLE Fill Missing File Name
196             ********************************************************************
197             ********************************************************************
198             **
199             ** Name:(S) FILFIL  -  Fill in Missing File Name
200             **
201             ** Category:   FILUTL
202             **
203             ** Purpose:
204             **      Adjusts file spec info on Save Stack to fill in missing
205             **      file name if necessary.  If the destination file name
206             **      is null, it always receives the source file name.  If
207             **      source file name is null, it receives desination file
208             **      name unless source device is CARD or PCRD, or if high
209             **      bit of the device info is set.  Status is returned
210             **      indicating if one file spec (or both) is external, and
211             **      if both file names are undefined.
212             **
213             ** Entry:
214             **      P       = 0
215             **      File specs on Save Stack as per SVINFO.
216             **      Upper nib of device info on stack has upper bit set if
217             **      source file name fill is NOT to be done for this file
218             **      spec.
219             **
220             ** Exit:
221             **      P       = 0
222             **      Updated file specs on Save stack as per SVINFO, with
223             **        the no-fill flag cleared for each file spec
224             **      S(sEXTDV) =  Set if either or both file specs are
225             **                   on HPIL device.
226             **      S(sUNDEF) =  Set if both file names are zero
227             **                   (that is, undefined).
228             **      S(sCARD)  =  1 if Source or Dest Device = CARD|PCRD
229             **      S(sDEST)  =  0 ("Source")
230             **      A         =  First 8 chars of source file name
231             **      R0(3-0)   =  Last 2 chars of source file name
232             **      D(A)      =  Source device info from RDINFO
233             **      R2(A)     =  Dest device info from RDINFO
234             **      Carry     =  Clear
235             **
236             ** Calls:      RDINFS, RDINFD, SVINFO, MFDEVC, MFDVC-
237             **
238             ** Uses.......
239             **  Inclusive: A,B,C,D(A),D1,R0,R1,R2,S4-S0
240             **
241             ** Stk lvls:   2
242             **
243             ** Detail:     Module Flow:
244             **             -----------
245             **             Clear Status
246             **             Read Source info, check device type and save away
247             **             Read Dest info
248             **             If Source file is undefined and device not card
249             **                Source file name <-- Dest file name
```

```
250                **              Check Dest device type
251                **              If Dest file name is undefined
252                **                 and neither device is CARD | PCRD
253                **                   Dest file name <-- Source file name
254                **              Write back Dest file info
255                **              Recall Source file info
256                **              Check Source device type
257                **              Write back Source file info
258                **
259                ** History:
260                **
261                **    Date      Programmer            Modification
262                **  --------    ----------   ----------------------------------
263                **  05/15/82       FH        Designed and coded.
264                **  02/15/83       FH        Added check for "No fill" bit of
265                **                             device code
266                **
267                ********************************************************************
268                ********************************************************************
269 011CE 840   =FILFIL ST=0    =sEXTDV
270 011D1 841          ST=0    =sUNDEF
271 011D4 8E00         GOSUBL  =RDINFS        Read source info and save away
           00
272 011DA 7701         GOSUB   MFDEVC         Check source device type
273 011DE AF8          B=A     W               .
274 011E1 118          C=R0                    .
275 011E4 109          R1=C                    .
276 011E7 DB           C=D     A               . (device code)
277 011E9 10A          R2=C                    .
278 011EC 8E00 FIL010  GOSUBL  =RDINFD        Read Dest info
           00
279 011F2 8AD          ?B#0    A              Source file defined?
280 011F5 81           GOYES   FIL020
281 011F7 872          ?ST=1   =sCARD         Source device CARD or PCRD?
282 011FA 31           GOYES   FIL020
283 011FC 11A          C=R2                   Recall source device code
284 011FF C6           C=C+C   A              No fill bit set?
285 01201 4B0          GOC     FIL020         GOYES
286 01204 AF8          B=A     W              Source file name <-- Dest file name
287 01207 118          C=R0                    .
288 0120A 109          R1=C                    .
289 0120D 77D0 FIL020  GOSUB   MFDVC-         Check dest device, set status
290 01211 8AC          ?A#0    A              Dest file defined?
291 01214 B0           GOYES   FIL030
292 01216 AF4          A=B     W              Dest file name <-- Source file name
293 01219 119          C=R1                    .
294 0121C 108          R0=C                    .
295 0121F 90B  FIL030  ?D=0    P              Device NOT :MAIN?
296 01222 A0           GOYES   FIL035
297 01224 7A30         GOSUB   FIL050         Convert dest filename to upper case
298 01228 7630         GOSUB   FIL050          .
299 0122C D6   FIL035  C=A     A              Test for both files undefined
300 0122E 0EFD         C=C!B   A               .
301 01232 8AE          ?C#0    A              Either file defined?
302 01235 50           GOYES   FIL040
```

```
303 01237 851          ST=1   =sUNDEF        Set status
304 0123A 7310 FIL040  GOSUB  FIL045         Write back destination file info
305 0123E 843          ST=0   =sDEST          Write back source file info
306 01241 AF4          A=B    W               .
307 01244 119          C=R1                   .
308 01247 108          R0=C                   .
309 0124A DB           C=D    A               . (swap dest, source dev codes)
310 0124C 12A          CR2EX                  .
311 0124F D7           D=C    A               .
312 01251 34FF FIL045  LCHEX  7FFFF           . Clear no-fill flag
          FF7
313 01258 0EF3         D=C&D  A               . .
314 0125C 8C00 svinfo  GOLONG =SVINFO         . (and return carry clear)
          00
315 01262 120  FIL050  AROEX                  Convert AROEX to upper case
316 01265 8C00         GOLONG =CVUCW          .
          00
```

```
317           FLADDR  STITLE Find First/Last Address
318           ************************************************************
319           ************************************************************
320           **
321           ** Name:(S) FLADDR  -  Find First/Last Address of Mem Device
322           **
323           ** Category:   FILUTL
324           **
325           ** Purpose:
326           **      Find the first and last address of available memory on
327           **      the specified memory device (PORT or MAIN).
328           **
329           ** Entry:
330           **      D(S)    =  Device type code of memory device (MAIN = 0,
331           **                   IRAM = 1, ROM = 2, etc)
332           **      D(0)    =  Port number if PORT device
333           **      D(1)    =  Extender number if PORT device
334           **      D(7-2)  =  Nibs 8-3 on configuration table entry for
335           **                 port device (contains size, address)
336           **      P       =  0
337           **
338           ** Exit:
339           **      A(A)    =  Address of first nib available memory on
340           **                 device
341           **      C(A)    =  Address of last nib available memory on
342           **                 device
343           **      D       =  Entry state
344           **      D1      @  AVMEMS for MAIN device
345           **              =  Size of module if PORT device
346           **      P       =  0
347           **      Carry clear
348           **
349           ** Calls:      EOFLC+, LSTADR
350           **
351           ** Uses.......
352           **  Exclusive: A(A),C,D1
353           **  Inclusive: A,   C,D1
354           **
355           ** Stk lvls:   2
356           **
357           ** Algorithm:
358           **      If PORT then
359           **         Start of module plus offset to file chain
360           **         Skip to end of file chain
361           **         Space beyond chain to av mem start
362           **         Find last address (call LSTADR)
363           **      Else (it's MAIN)
364           **         Fetch AVMEMS, AVMEME
365           **
366           ** History:
367           **
368           **    Date      Programmer             Modification
369           **  --------    ----------   ----------------------------------
370           **  06/11/82       FH        Designed and coded
371           **
```

```
372                ********************************************************************
373                ********************************************************************
374 0126B 94B  =FLADDR ?D=0   S              System RAM?
375 0126E 12           GOYES  FLAD20
376 01270 AFB          C=D    W              Find address of file chain offset
377 01273 BF6          CSR    W               .
378 01276 3180         LC(2)  8               .
379 0127A 8E00         GOSUBL =EOFLC+         Skip to end of file chain
          00
380 01280 171          D1=D1+ 2               . and space to av. mem. start
381 01283 8F00         GOSBVL =LSTADR          .
          000
382 0128A 133          AD1EX
383 0128D 03           RTNCC
384 0128F 1F99  FLAD20 D1=(5) =AVMEME         Set up exit conditions
          5F2
385 01296 147   FLAD30 C=DAT1 A
386 01299 1F49 =D1@AVS D1=(5) =AVMEMS         For documentation see OBCOLL module
          5F2
387 012A0 143          A=DAT1 A
388 012A3 03           RTNCC
```

```
389             MEMCKL  STITLE MEMCKL - Memory Check with Leeway
390             ************************************************************
391             ************************************************************
392             **
393             ** Name:(S) MEMCKL  -  Check Avail Memory With, Without Leeway
394             ** Name:    MEMCK+  -  Check Avail Memory With, Without Leeway
395             ** Name:    CHKSPC  -  Check Available Memory With Leeway
396             ** Name:    CHKSPF  -  Check Available Memory Without Leeway
397             ** Name:(S) CHKmem  -  Check Available Memory Without Leeway
398             **
399             ** Category:   PTRUTL
400             **
401             ** Purpose:
402             **      See if requested memory amount [+ Leeway] is less than
403             **      or equal to available memory.  Nonzero value of P on
404             **      entry determines whether leeway will be included in
405             **      check for some entry points.  "Insufficient Memory"
406             **      error code is returned with carry set if requested
407             **      amount exceeds the available memory.
408             **
409             ** Entry:
410             **    MEMCKL:
411             **      C(A)   = Absolute amount memory to check
412             **      P      = 0 iff LEEWAY to be added to amt being checked
413             **    MEMCL+:
414             **      B(A)   = Absolute amount memory to check
415             **      P      = 0 iff LEEWAY to be added to amt being checked
416             **    CHKSPC:  (LEEWAY ALWAYS added; B(A) not used)
417             **      C(A)   = Absolute amount memory to check
418             **      P      = 0
419             **    CHKSPF:  (LEEWAY NEVER added; B(A) not used)
420             **      C(A)   = Absolute amount memory to check
421             **      D1     @ Available memory end pointer
422             **    CHKmem:  (LEEWAY NEVER added; B(A) not used)
423             **      A(A)   = Available memory end
424             **      C(A)   = Absolute amount memory to check
425             **      P      = 0
426             **
427             ** Exit:
428             **    Carry Clear:   Enough memory
429             **      B(A)   = Amount to check  (MEMCKL, MEMCL+ only)
430             **      A(A)   = Available Memory start
431             **      D1     @ AVMEMS
432             **      C(A)   = Available memory MINUS requested amount
433             **               (MINUS Leeway if also checked)
434             **      P      = 0
435             **    Carry set:     Not enough memory
436             **      B(A)   = Amount to check  (MEMCKL, MEMCL+ only)
437             **      C(A)   = eMEM
438             **      P      = 0
439             **
440             ** Calls:      None
441             **
442             ** Uses.......
443             **  Inclusive: A(A),C(A),D1,B(A) (MEMCKL, MEMCL+ only)
```

```
444                **
445                ** Stk lvls:   0
446                **
447                ** Algorithm:
448                **
449                ** MEMCKL: B <-- Requested Amount
450                ** MEMCL+: C <-- B
451                **         If P=0
452                ** CHKSPC:     C <-- Leeway
453                **             Amount = Req Amount + Leeway
454                **             If overflow --->  Error Return
455                **         D1 <-- AVMEME
456                ** CHKSPF: A <--  Available Memory End
457                ** Chkmem: Subtract Req Amount from Available Memory End
458                **         If negative  -->     Error Return
459                **         D1 <-- AVMEMS
460                **         A <--  Available Memory Start
461                **         Subtract Avail Memory start from subtracted amount
462                **         If negative, then
463                **            Error Return   [ C <-- eMEM ]
464                **         else
465                **            Return carry clear
466                **
467                ** History:
468                **
469                **    Date      Programmer            Modification
470                **  --------    ----------   ----------------------------------
471                **  07/04/82       JP        Modified documentation
472                **  09/11/82       JP        Added Leeway check code
473                **  10/24/83       FH        Updated documentation
474                **
475                ********************************************************************
476                ********************************************************************
477 012A5 D5    =MEMCKL B=C     A              Save allocation amount in B(A)
478 012A7 D9    =MEMCL+ C=B     A              Move amount to check into C(A)
479 012A9 880           ?P#     0              NO Leeway check?
480 012AC F0            GOYES   CKSPC1          .
481 012AE DA    =CHKSPC A=C     A              Add leeway count to allocation
482 012B0 D2            C=0     A               .
483 012B2 314D          LC(2)   =LEEWAY         .
484 012B6 C2            C=A+C   A               .
485 012B8 422           GOC     CKSPC2         If overflow, error out
486 012BB 1F99  CKSPC1  D1=(5)  =AVMEME        Set D1 to avail memory end ptr
          5F2
487 012C2 20    =CHKSPF P=      0              Reset P to 0
488 012C4 143           A=DAT1  A              Read available memory end
489 012C7 EE    =CHKmem C=A-C   A              Compute [End] - [Requested amt]
490 012C9 411           GOC     CKSPC2         Error if negative (carry set)
491 012CC 1F49          D1=(5)  =AVMEMS        D1 @ Av memory start (DON'T pack)
          5F2
492 012D3 143           A=DAT1  A              Read available memory start
493 012D6 E2            C=C-A   A              Compute [End] - [Amt] - [Start]
494 012D8 500           RTNNC                  Return OK if not negative
495 012DB 6311  CKSPC2  GOTO    RMEM10         Load memory error number
```

```
MFDEVC  STITLE Mainframe Device Check
*****************************************************************
*****************************************************************
**
** Name:    MFDEVC  -  Mainframe Device Check on File Info
** Name:    MFDVC-  -  Mainframe Device Check on File Info
** Name:    MFDVC+  -  Mainframe Device Check on Selected File
**
** Category:   FILUTL
**
** Purpose:
**      Indicate whether device is in mainframe, on card, or on
**      HPIL.  MFDVC+ calls RDINFO before the check is made.
**      MFDEV- performs like MFDEVC, but will leave sCARD (S2)
**      UNTOUCHED if the device is not CARD or PCRD.
**
** Entry:
**      P       =  0
**      Source, dest file info on the Save Stack as per SVINFO
**     MFDEVC, MFDEVC-:
**      D(A)    =  Device code as per RDINFO
**     MFDVC+:
**      S3(sDEST) =  1 if destination file info is to be read
**                     before falling into MFDEVC
**                =  0 if destination file info is to be read
**                     before falling into MFDEVC
**
** Exit:
**      S2(sCARD)  =  Set if device is CARD or PCRD (all entry
**                    pts), else it is CLEARED (MFDEVC and
**                    MFDEV+ only)
**      S0(sEXTDV) =  Set if device is HPIL,
**                    else it is UNCHANGED
**      D(A)    =  Device code from RDINFO (NOTE: This code
**                 has been shifted one nibble left circular
**                 from where FSPECx originally returned it)
**      C(0)    =  7 (CARD device code, =dCARD)
**      C(4-1)  =  D(4-1) exit condition
**      P       =  0
**      Carry   =  Clear
**    MFDVC+ also exits with:
**      A       =  First 8 characters of file name
**      R0      =  Last 2 characters of file name
**
** Calls:      RDINFO.
**
** Uses.......
**  Exclusive:  C(A),D(XS),    sCARD,sEXTDV,sDEST,sREADI
**  Inclusive:  A,C,D(A),D1,R0,S0,S2,S3,S4
**
** Stk lvls:   1
**
** History:
**
**    Date     Programmer             Modification
```

```
551             **  --------   ----------   ----------------------------------
552             **  11/15/82       FH       Modified documentation, supplied
553             **                            C(0) exit condition
554             **
555             ********************************************************************
556             ********************************************************************
557 012DF 8E00  =MFDVC+ GOSUBL =RDINFO        Read device info
          00
558 012E5 842   =MFDEVC ST=0    =sCARD        Clear Card status
559 012E8 DB    =MFDVC- C=D     A             Fetch device info
560 012EA B06           C=C+1   P             Device undefined?
561 012ED 4D0           GOC     MFD010        GOYES
562 012F0 DB            C=D     A
563 012F2 A06           C=C+C   P             Not HPIL device?
564 012F5 550           GONC    MFD010         . GOYES
565 012F8 850           ST=1    =sEXTDV
566 012FB 307   MFD010  LC(1)   dCARD          . (load up card device code)
567 012FE 907           ?D#C    P             Device not CARD or PCRD?
568 01301 50            GOYES   MFD020
569 01303 852           ST=1    =sCARD        Set CARD Device flag
570 01306 03    MFD020  RTNCC
```

```
571            MOVE*M  STITLE Move Memory Up or Down
572            ************************************************************
573            ************************************************************
574            **
575            ** Name:(S) MOVE*M  -  Move Memory Up or Down Without Ref Adj
576            **
577            ** Category:   GENUTL
578            **
579            ** Purpose:
580            **      Move memory up or down with no reference adjust.
581            **
582            ** Entry:
583            **      A(A)   =  Source address
584            **      B(A)   =  Length of block to move in nibs
585            **      C(A)   =  Dest address
586            **
587            ** Exit:
588            **      All entry conditions
589            **      P      =  0
590            **
591            ** Calls:      MOVEDM, MOVEUM
592            **
593            ** Uses.......
594            **  Exclusive: A, C(A), D0, D1
595            **  Inclusive: A, C(A), D0, D1, P
596            **
597            ** Stk lvls:   1
598            **
599            ** History:
600            **
601            **    Date     Programmer              Modification
602            **  --------   ----------   ----------------------------------
603            **  06/14/82      FH        Designed and coded.
604            **
605            ************************************************************
606            ************************************************************
607 01308 DE   =MOVE*M ACEX   A
608 0130A 8B2          ?A<C   A              Move to lower address?
609 0130D 51           GOYES  MOVE10
610 0130F C0           A=A+B  A              Compute end of dest
611 01311 C9           C=C+B  A              Compute end of source
612 01313 8F00         GOSBVL =MOVEDM
          000
613 0131A 132          ADOEX                  .
614 0131D 137          CD1EX                 Set up exit conditions
615 01320 03           RTNCC
616 01322 8F00  MOVE10 GOSBVL =MOVEUM
          000
617 01329 132          ADOEX                  .
618 0132C E0           A=A-B  A               .
619 0132E 137          CD1EX                 Set up exit conditions
620 01331 E9           C=C-B  A               .
621 01333 03           RTNCC
```

```
622             MVMEM   STITLE Move File Memory W/Ref Adjust
623             ********************************************************************
624             ********************************************************************
625             **
626             ** Name:    MVMEM   -  Move File Memory W/Ref Adjust
627             ** Name:(S) MVMEM+  -  Move File Memory W/Ref Adjust
628             **
629             ** Category:   GENUTL
630             **
631             ** Purpose:
632             **      Move memory in a file chain up or down with reference
633             **      adjust.  Works for either MAIN or Independent RAM.
634             **      RFADJ is called, and pointers MAINEN -> AVMEMS and
635             **      CURRST -> CURREN are updated if they fall into the
636             **      block that moved.  Note that if the pointer value falls
637             **      outside the block that moved but inside the area into
638             **      which it moved, no action is taken.  If the source of
639             **      the move is NOT EQUAL to the corresponding file header
640             **      address passed in C(A), then that file header's chain
641             **      length is also adjusted.
642             **
643             ** Entry:
644             **      A(A)   =  Starting address to move up or down.  Equal
645             **                to C(A) if adding or deleting file to/from
646             **                file chain.
647             **      B(A)   =  Offset (dest address - source address)
648             **      C(A)   =  Address of header of file containing address
649             **                to be moved.  File chain length field of the
650             **                header will be updated to new length if and
651             **                only if C(A) # A(A).  If adding or deleting a
652             **                file to or from the chain, this address should
653             **                point to the following file header in the file
654             **                chain or to the end of the chain.
655             **      P      =  0
656             **     MVMEM:
657             **      D(S)   =  Device code for memory device
658             **      D(B)   =  Port number if port device
659             **      D(7-2) =  Nibs 8-3 of port's configuration table entry
660             **     MVMEM+:
661             **      D entry state will be computed from C(A)
662             **
663             ** Exit:
664             **      R0     =  A(A) entry: starting address of move
665             **      R2     =  C(A) entry: start of file header
666             **      B(A)   =  Entry state
667             **      P      =  0
668             **     Carry clear:
669             **      Memory moved and references adjusted
670             **     Carry set:
671             **      C(3-0) =  Error code if error occurred:
672             **                  eMEM   - Insufficient Memory
673             **                  eILACS - Illegal Access (if ROM or EPROM)
674             **
675             **
676             ** Calls:      LOCADR, FLADDR, RMEMCH, MOVE*M, ADJREF
```

```
677                **
678                ** Uses.......
679                **  Exclusive: A,B,C,D,D0,D1,R0,R1,R2
680                **  Inclusive: A,B,C,D,D0,D1,R0,R1,R2,SCRTCH(4-0)
681                **
682                ** Stk lvls:   3
683                **
684                ** NOTE:
685                **      NO CHECK IS MADE to verify that the starting address
686                **      actually falls within a file chain or whether the port
687                **      specified corresponds to the specified address.
688                **
689                ** Algorithm:
690                **   MVMEM+ :
691                **      Compute memory device info
692                **   MVMEM  :
693                **      If move is memory expansion then
694                **         Check memory (return if error)
695                **      If source # file header start then
696                **         Update chain length
697                **      Move memory
698                **      Adjust references
699                **
700                ** History:
701                **
702                **     Date      Programmer            Modification
703                **   --------   ----------   ---------------------------------
704                **  06/09/82       FH        Designed and coded.
705                **                 SW        Check for ROM file
706                **  02/15/83       FH        Packed, updated documentation
707                **
708                ********************************************************************
709                ********************************************************************
710 01335 8D00  MVME00 GOVLNG =PRGFE          Note: C(S), D(S) have meaning!
           000
711                *                          Loads eFACCS in C(3-0), RTNSC.
712                *
713 0133C 134  =MVMEM+ D0=C                    Save file header address
714 0133F D9           C=B    A              Save offset in R0
715 01341 108          R0=C                   .
716 01344 D6           C=A    A              Get memory device info
717 01346 8F00         GOSBVL =LOCADR         .
           000
718 0134D ACB          C=D    S
719 01350 A4E          C=C-1  S
720 01353 470          GOC    MVME20          MAIN?
721 01356 94E          ?C#0   S               ROM or
722 01359 CD           GOYES  MVME00          ROM or PROM?
723 0135B 118   MVME20 C=R0                   Restore offset
724 0135E D5           B=C    A               .
725 01360 136          CDOEX                  Restore file header address
726 01363 100  =MVMEM  R0=A                   Save source start for RFADJ in R0
727 01366 10A          R2=C                   Save file header address    in R2
728 01369 AC1          B=0    S               Presume "Memory expansion"
729 0136C D6           C=A    A              Compute dest address in C
```

```
730 0136E C9          C=C+B  A              .
731 01370 8B2         ?C>A   A              Memory expansion?
732 01373 A0          GOYES  MVME40
733 01375 8A2         ?A=C   A              Nothing to move?
734 01378 E8          GOYES  MFD020         RTNCC
735 0137A B45         B=B+1  S              Flag "Memory contraction"
736 0137D 7AEE  MVME40 GOSUB =FLADDR        Compute dev first & last address
737 01381 94D         ?B#0   S              Memory contraction?
738 01384 90          GOYES  MVME60
739 01386 7340        GOSUB  =RMEMCH        Check available memory
740 0138A 400         RTNC
741             *
742             *  At this point A is at av mem start of port or MAIN
743             *
744 0138D 120   MVME60 AROEX                A = SOURCE, R0 = av mem st
745 01390 11A         C=R2                  Fetch file hdr addr
746 01393 8A2         ?A=C   A              Source = file header?
747 01396 60          GOYES  MVME80         If so, don't update file header
748             *
749             *  Update file chain length
750             *
751 01398 7872        GOSUB  UPDFCL
752             *
753             *  Move memory
754             *
755 0139C 118   MVME80 C=R0                 B = LEN  = av mem st - source
756 0139F E2          C=C-A  A               .
757 013A1 DD          BCEX   A               .
758 013A3 C2          C=A+C  A              C = DEST = source + offset
759 013A5 7F5F        GOSUB  =MOVE*M        Move memory
760             *
761             *  Adjust references
762             *  by falling into ADJREF
763             *
```

```
                ADJREF  STITLE Adjust All Memory References
                ********************************************************************
                ********************************************************************
                **
                ** Name:    ADJREF  -  Adjust All Memory References
                **
                ** Category:   PTRUTL
                **
                ** Purpose:
                **      ADJREF adjusts all pointers (CURRST -> AVMEMS) and
                **      memory references if they fall within the specified
                **      block that moved.  The memory may be up or down, and
                **      in either MAIN or IRAM memory.
                **
                ** Entry:
                **     ADJREF:    [System RAM or IRAM Move]
                **      A(A)   =  Source start of move
                **      B(A)   =  Length of block that moved in nibs
                **      C(A)   =  Destination start of move
                **
                ** Exit:
                **      R0     =  Source start of move
                **      B(A)   =  Entry state
                **      P      =  0
                **
                ** Calls:      RFAD+I, RFAD-I
                **
                ** Uses.......
                **  Inclusive: A,B,C,D,D0,D1,R0,R1,R2,P,SCRTCH(4-0)
                **
                ** Stk lvls:   2
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------   ----------------------------------
                **  10/11/82      FH        Adapted from RPLLIN code
                **
                ********************************************************************
                ********************************************************************
                *
                *  NOTE:  The above code of MVMEM falls into here
                *
 013A9 100  =ADJREF R0=A                  Save source start for RFADJ
 013AC DC           ABEX   A              A(A)=Source len;B(A)=Source start
 013AE 1F10         D1=(5) =SCRTCH
       9F2
 013B5 C0           A=A+B  A              A(A)=End Source
 013B7 141          DAT1=A A
 013BA ED           B=C-B  A              Offset: Dest Start-Source Start
 013BC 490          GOC    ADJR20         Carry => Mem moving to lower addr
 013BF 8D00         GOVLNG =RFAD+I        Use same entry pt for MAIN & IRAM
       000
 013C6 8D00  ADJR20 GOVLNG =RFAD-I        Use same entry pt for MAIN & IRAM
       000
```

```
                RMEMCH  STITLE ROM/RAM Memory Check
                ********************************************************************
                ********************************************************************
                **
                ** Name:    RMEMCH  -  Check Memory in Memory Device
                **
                ** Category:   GENUTL
                **
                ** Purpose:
                **      Check if there is a specified amount of room in the
                **      specified memory device given its first and last
                **      address of available memory; return error code if not.
                **      If the device is MAIN, the memory check is done with
                **      LEEWAY.
                **
                ** Entry:
                **      A(A)    =  Address of first nib of available memory on
                **                 the device
                **      B(A)    =  Amount of memory requested
                **      C(A)    =  Address of last nib of available memory on
                **                 the device + 1
                **      P       =  0
                **
                ** Exit:
                **      A(A)    =  Entry state
                **      B(A)    =  Entry state
                **      P       =  0
                **     Carry clear:
                **      C(A)    =  Total memory available on this device
                **     Carry set:
                **      C(3-0)  =  Error code "Insufficient Memory" (eMEM)
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: C(A), D(A)
                **
                ** Stk lvls:   None
                **
                ** NOTE:
                **      LEEWAY is included in the memory check iff MAIN device.
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ---------------------------------
                **  06/11/82       FH        Designed and coded.
                **  03/10/83       FH        Check for < 0 memory
                **
                ********************************************************************
                ********************************************************************
 013CD E2    =RMEMCH C=C-A  A               Compute available memory
 013CF 8B1           ?C<B   A               Not enough room?
 013D2 D1            GOYES  RMEM10
 013D4 94F           ?D#0   S               Not in MAIN?
```

```
871 013D7 61          GOYES  RMEM05
872             *
873             * Check memory with LEEWAY if in MAIN
874             *
875 013D9 D7          D=C    A
876 013DB D2          C=0    A
877 013DD 314D        LC(2)  =LEEWAY
878 013E1 DF          CDEX   A
879 013E3 EB          C=C-D  A
880 013E5 490         GOC    RMEM10         If less than 0 memory
881 013E8 8B1         ?C<B   A
882 013EB 40          GOYES  RMEM10
883             *
884 013ED 03   RMEM05 RTNCC
885 013EF 3300 =RMEM10 LC(4)  =eMEM          Return error code
          00
886 013F5 02          RTNSC
```

```
887             RPLLIN  STITLE Replace Line in Memory File
888             ********************************************************************
889             ********************************************************************
890             **
891             ** Name:(S) RPLLIN  -  Replace Line in Memory File
892             **
893             ** Category:   FILUTL
894             **
895             ** Purpose:
896             **      Replace a line in a memory file with the contents of
897             **      the output buffer.  May be used to insert, delete, or
898             **      replace a line in the file.
899             **
900             ** Entry:
901             **      OUTBS  @  Start of replacement line
902             **      AVMEMS @  End of replacement line (address of last
903             **                nib + 1)
904             **      A(A)   =  Address of last nib + 1 of old line
905             **      C(A)   =  Address of file header of file
906             **      R3(A)  =  Length of OLD line in nibs  (zero for
907             **                insertion)
908             **      P      =  0
909             **
910             ** Exit:
911             **      R3(A)  =  Offset of move (DEST END - SOURCE END)
912             **      P      =  0
913             **     Carry clear:  [Successful replacement]
914             **      Output buffer collapsed
915             **      A(A)   =  End + 1 of replaced line in file
916             **      B(A)   =  Length of replacement line in nibs
917             **      C(A)   =  (OUTBS)
918             **     Carry set:
919             **      C(3-0) =  Error code:
920             **                  eMEM   - Insufficient memory
921             **                  eILACS - Illegal access (if ROM or PROM)
922             **
923             ** Calls:      OBLCMP, MOVE*M, MVMEM+, INITPT
924             **
925             ** Uses.......
926             **  Exclusive: A,B(A),C,                D1,R0,R1,   R3
927             **  Inclusive: A,B   ,C,D(S),D(7-0),D0,D1,R0,R1,R2,R3
928             **
929             ** Stk lvls:   3
930             **
931             ** NOTE:
932             **      Security and privacy are not checked.  ROM or EPROM
933             **      access returns eFACCS error.
934             **
935             **
936             ** Algorithm:
937             **
938             ** History:
939             **
940             **     Date    Programmer             Modification
941             **   --------  ----------   ----------------------------------
```

```
942                **                FH        Adapted from a TRANSFORM utility
943                **  02/15/83      FH        Packed and updated documentation
944
945                **
946                ****************************************************************
947                ****************************************************************
948 013F7 108   =RPLLIN R0=C                    Save file header
949 013FA 101          R1=A                    Save line end + 1
950 013FD 7B80         GOSUB  OBLCMP           Compute line length
951 01401 D8           B=A    A                 . into B
952 01403 123          AR3EX                   Exchange old line, new line len
953 01406 E8           B=B-A  A                B = OFFSET
954 01408 118          C=R0                    C = FILE HEADER
955 0140B 111          A=R1                    A = SOURCE = old line end + 1
956 0140E 7A2F         GOSUB  MVMEM+           Move memory out
957 01412 D4           A=B    A                R3 = offset
958 01414 123          AR3EX                   . and A = New line len
959 01417 400          RTNC                    Return if error
960 0141A 118          C=R0                    Recall old line end + 1
961 0141D C9           C=C+B  A                C = new line end + 1
962 0141F E2           C=C-A  A                C = DEST = old line begin
963 01421 D8           B=A    A                B = LEN
964 01423 1FF8         D1=(5) =OUTBS           A = SOURCE = (OUTBS)
          5F2
965 0142A 143          A=DAT1 A                 .
966 0142D 77DE         GOSUB  MOVE*M           Move new line into position
967 01431 C9           C=C+B  A                A = New line end
968 01433 DA           A=C    A                 .
969                ****************************************
970                *  NOTE:  FALLS INTO FOLLOWING CODE  *
971                ****************************************
```

```
OBCOLL  STITLE Output Buffer Pointer Manipulation
*****************************************************************
*****************************************************************
**
** Name:     CLCOLL  -  Collapse Buffer Pointers to CLCSTK
** Name:     SYCOLL  -  Collapse Buffer Pointers to SYSEN
** Name:(S) OBCOLL  -  Collapse Output Buffer
** Name:     BBCOLL  -  Collapse Input, Output Buffer Pointers
** Name:     OBPRD   -  Read Output Buffer Pointers
** Name:     OBLCMP  -  Compute Output Buffer Length
** Name:     INBS=C  -  Set INBS to the Value in C
** Name:     D1=IBS  -  Set D1 to Start of Input Buffer
** Name:(S) D1@AVS  -  Set D1 to Available Memory Start
** Name:     AVS=D0  -  Set AVMEMS to Value in D0
** Name:     AVS=C   -  Set AVMEMS to Value in C
**
** Category:   PTRUTL
**
** Purpose:
**      Manipulate buffer pointers.
**
**    CLCOLL:
**      Collapse SYSEN, OUTBS, and AVMEMS to CLCSTK.
**
**    SYCOLL:
**      Collapse OUTBS and AVMEMS to SYSEN.
**
**    OBCOLL:
**      Collapse AVMEMS to OUTBS (collapse output buffer).
**
**    BBCOLL:
**      Collapse INBS, OUTBS, and AVMEMS to SYSEN (collapse
**      both input and output buffers).
**
**    OBPRD:
**      Read output buffer pointers OUTBS and AVMEMS into C(A),
**      A(A).
**    OBLCMP:
**      Compute length of output buffer = (AVMEMS) - (OUTBS).
**
**    INBS=C:
**      Set INBS to the value in C.
**
**    D1=IBS:
**      Set D1 to start of input buffer.
**
**    D1@AVS:
**      Set D1 to AVMEMS, A(A) to (AVMEMS).
**
**    AVS=D0:
**      Set AVMEMS to value of D0.
**
**    AVS=C:
**      Set AVMEMS to the value in C(A).
**
```

```
1027            **
1028            ** Entry:
1029            **    No entry conditions assumed unless explicitly stated below.
1030            **
1031            **    INBS=C:
1032            **      C(A)   =  Value to store in INBS.
1033            **
1034            **    AVS=C:
1035            **      C(A)   =  Value to store in AVMEMS.
1036            **
1037            **
1038            ** Exit:
1039            **
1040            **    CLCOLL:
1041            **      C(A)   =  (CLKSTK)
1042            **      D1     =  5 beyond AVMEMS
1043            **      Carry  =  Clear
1044            **
1045            **    SYCOLL:
1046            **      C(A)   =  (SYSEN)
1047            **      D1     =  5 beyond AVMEMS
1048            **      Carry  =  Clear
1049            **
1050            **    OBCOLL:
1051            **      C(A)   =  (OUTBS)
1052            **      D1     =  5 beyond AVMEMS
1053            **      Carry  =  Clear
1054            **
1055            **    BBCOLL:
1056            **      C(A)   =  (SYSEN)
1057            **      D1     =  INBS
1058            **      Carry  =  Clear
1059            **
1060            **    OBPRD:
1061            **      A(A)   =  (AVMEMS)
1062            **      C(A)   =  (OUTBS)
1063            **      D1     @  AVMEMS
1064            **      Carry  =  Clear
1065            **
1066            **    OBLCMP:
1067            **      A(A)   =  Length of output buffer -- (AVMEMS) - (OUTBS)
1068            **      C(A)   =  (OUTBS)
1069            **      D1     @  AVMEMS
1070            **      Carry  =  Clear
1071            **
1072            **    INBS=C:
1073            **      C(A)   =  Entry state
1074            **      D1     =  INBS
1075            **      Carry  =  Clear
1076            **
1077            **    D1=IBS:
1078            **      D1     @  Start of input buffer
1079            **      C(A)   =  INBS
1080            **      Carry preserved
1081            **
```

```
1082                **    AVS=DO:
1083                **      C(A)   =  AVMEMS
1084                **      Carry  =  Clear
1085                **
1086                **    AVS=C:
1087                **      C(A)   =  AVMEMS
1088                **      DO     @  C(A) entry value
1089                **      Carry  =  Clear
1090                **
1091                **
1092                ** Calls:      INITPT  (CLCOLL,SYCOLL,OBCOLL only)
1093                **
1094                ** Uses.......
1095                **  Inclusive: C(A),D1       (CLCOLL,SYCOLL,OBCOLL,BBCOLL,
1096                **                             INBS=C,D1=IBS)
1097                **             A(A),D1       (D1@AVS)
1098                **             A(A),C(A),D1  (OBPRD,OBLCMP)
1099                **             C(A)          (AVS=DO)
1100                **             C(A),DO       (AVS=C)
1101                **
1102                ** Stk lvls:   0 (CLCOLL,SYCOLL,OBCOLL,OBPRD,INBS=C,
1103                **                D1=IBS,D1@AVS,AVS=DO,AVS=C)
1104                **             1 (OBLCMP)
1105                **             2 (BBCOLL)
1106                **
1107                ** History:
1108                **
1109                **    Date     Programmer             Modification
1110                **   --------  ----------   ---------------------------------
1111                **  09/16/82      FH        Designed and coded.
1112                **  10/12/82      FH        Added CLCOLL,SYCOLL,BBCOLL,INBS=C,
1113                **                            D1=IBS,AVS=DO,AVS=C
1114                **  02/10/83      FH        Removed IBPRD,OBSKIP,OBBACK
1115                **
1116                ********************************************************************
1117                ********************************************************************
1118                ************************************
1119                *  NOTE:  FALLEN INTO FROM ABOVE !!  *
1120                ************************************
1121 01435 1FF8 =OBCOLL D1=(5) =OUTBS         Fetch Output Buffer Start
           5F2
1122 0143C 2E           P=     15-((AVMEMS)-(OUTBS))/5
1123 0143E 147   OBCOL1 C=DAT1 A              Read value to collapse to
1124 01441 8D00 =initpt GOVLNG =INITPT        Set all pointers to C(A)
           000
1125                *
1126 01448 1F58 =CLCOLL D1=(5) =CLCSTK        Fetch Calc Stack start
           5F2
1127 0144F 2C           P=     15-((AVMEMS)-(CLCSTK))/5
1128 01451 6CEF         GOTO   OBCOL1
1129                *
1130 01455 1FA8 =SYCOLL D1=(5) =SYSEN         Fetch System End
           5F2
1131 0145C 2D           P=     15-((AVMEMS)-(SYSEN))/5
1132 0145E 6FDF         GOTO   OBCOL1
```

```
1133                *
1134 01462 7FEF    =BBCOLL GOSUB  SYCOLL          Collapse OUTBS,AVMEMS
1135 01466 1F6C    =INBS=C D1=(5) =INBS           Update INBS
           6F2
1136 0146D 145             DAT1=C A                .
1137 01470 03              RTNCC
1138                *
1139 01472 1F6C    =D1=IBS D1=(5) =INBS           Read INBS
           6F2
1140 01479 147             C=DAT1 A                .
1141 0147C 137             CD1EX                   . into D1
1142 0147F 01              RTN
1143                *
1144 01481 1FF8    =OBPRD  D1=(5) =OUTBS          C(A) = (OUTBS)
           5F2
1145 01488 6D0E            GOTO   FLAD30
1146                *
1147 0148C 71FF    =OBLCMP GOSUB  OBPRD           Read pointers
1148 01490 EA              A=A-C  A                Compute length
1149 01492 03              RTNCC
1150                *
1151 01494 136     =AVS=D0 CD0EX                  C(A) = new avmems
1152 01497 1B49    =AVS=C  D0=(5) =AVMEMS         Update AVMEMS
           5F2
1153 0149E 144             DAT0=C A                .
1154 014A1 136             CD0EX                   D0   = new avmems
1155 014A4 03              RTNCC
```

```
1156            RSTK<R  STITLE Restore RSTK Level(s) From RAM
1157            ******************************************************************
1158            ******************************************************************
1159            **
1160            ** Name:(S) RSTK<R  -  Restore RSTK Level(s) From RSTKBF Buffer
1161            **
1162            ** Category:   SAVUTL
1163            **
1164            ** Purpose:
1165            **      Restore Return Stack Level(s) from circular buffer.
1166            **      Levels are saved and restored on a last-in-first-out
1167            **      (LIFO) basis (see R<RSTK for save routine).  The buffer
1168            **      holds up to 16 levels.  No more than 6 levels should be
1169            **      saved or retored in one call, however, since the return
1170            **      to the caller of RSTK<R requires one level.
1171            **
1172            ** Entry:
1173            **      P       =  n - 1, where n is number of levels to restore
1174            **                 (not counting return to caller of R<RSTK)
1175            **
1176            ** Exit:
1177            **      Carry   =  Clear
1178            **      P       =  0
1179            **      D0      @  RSTKBp RAM location
1180            **
1181            ** Calls:      RSTK>1
1182            **
1183            ** Uses.......
1184            **  Inclusive: C(A), C(S), B(A), D0
1185            **
1186            ** Stk lvls:   n  (n levels are ADDED to the stack on return)
1187            **
1188            ** NOTE:
1189            **      The addresses stored in the buffer are NOT updated by
1190            **      RFADJ.
1191            **
1192            ** Detail:
1193            **      The position in the circular buffer is indicated by
1194            **      the nibble =RSTKBp in System RAM, which points to the
1195            **      last position written.
1196            **
1197            **      During the routine:
1198            **           C(S)   =  Level counter (from P on entry)
1199            **           P      =  Circular buffer position (from =RSTKBp)
1200            **      These counters are set up by routine RSTK>1, which is
1201            **      shared by RSTK<R and R<RSTK.
1202            **
1203            ** History:
1204            **
1205            **    Date     Programmer             Modification
1206            **  --------   ----------   ----------------------------------
1207            **  09/14/82      FH        Designed and coded
1208            **  02/24/83      FH        Expanded buffer from 8 to 16 levels
1209            **
1210            ******************************************************************
```

```
1211             ****************************************************************
1212 014A6 22   =RST2<R P=     2              Restore 3 levels.
1213 014A8 FD   =RSTK<R B=-B-1 A              Clear carry: no pointer increment
1214 014AA 07           C=RSTK
1215 014AC 7450         GOSUB  RSTK>1         Set up counters and position
1216             *
1217             *  Recall levels
1218             *
1219 014B0 146   R>RS20 C=DAT0 A              Restore level
1220 014B3 06           RSTK=C                 .
1221 014B5 184          D0=D0- 5               Move position
1222 014B8 0D           P=P-1                  .
1223 014BA 560          GONC   R>RS40          .
1224 014BD 19B6         D0=(2) (=RSTKBF)+15*5   . (wrap around)
1225 014C1 A4E   R>RS40 C=C-1  S               Decrement level counter
1226 014C4 5BE          GONC   R>RS20          Loop if more
1227             *
1228             *  Save position, restore registers, and return
1229             *
1230 014C7 80F0  R>RS60 CPEX   0               Save position
1231 014CB 19F1         D0=(2) =RSTKBp         .
1232 014CF 15C0         DAT0=C 1               .
1233 014D3 20           P=     0
1234 014D5 D9           C=B    A               Recall return address
1235 014D7 06           RSTK=C                 Return
1236 014D9 03           RTNCC
```

```
1237            R<RSTK  STITLE Save RSTK Level(s) Into RAM
1238            ********************************************************************
1239            ********************************************************************
1240            **
1241            ** Name:(S) R<RSTK  -  Save RSTK Level(s) Into RSTKBF Buffer
1242            **
1243            ** Category:   SAVUTL
1244            **
1245            ** Purpose:
1246            **      Save Return Stack Level(s) in circular buffer.  Levels
1247            **      are saved and restored on a last-in-first-out (LIFO)
1248            **      basis (see RSTK<R for restore routine).  The buffer may
1249            **      hold up to 16 levels.  No more than 6 levels should be
1250            **      saved or retored in one call, however, since the return
1251            **      to the caller of R<RSTK requires one level.
1252            **
1253            ** Entry:
1254            **      P      =  n - 1, where n is number of levels to save
1255            **                (not counting return to caller of R<RSTK,
1256            **                which is not saved)
1257            **
1258            ** Exit:
1259            **      Carry  =  Clear
1260            **      P      =  0
1261            **      D0     @  RSTKBp RAM location
1262            **
1263            ** Calls:      RSTK>1
1264            **
1265            ** Uses.......
1266            **  Inclusive: B(A), C(A), C(S), D0  (R<RSTK)
1267            **
1268            ** Stk lvls:  -n  (n levels are REMOVED from stack on return)
1269            **
1270            ** NOTE:
1271            **      The addresses stored in the buffer are NOT updated by
1272            **      RFADJ.
1273            **
1274            ** Detail:
1275            **      The position in the circular buffer is indicated by the
1276            **      nibble =RSTKBp in System RAM, which points to the last
1277            **      position written.
1278            **
1279            **      During the routine:
1280            **           C(S)   =  Level counter (from P on entry)
1281            **           P      =  Circular buffer position (from =RSTKBp)
1282            **      These counters are set up by routine RSTK>1, which is
1283            **      shared by RSTK<R and R<RSTK.
1284            **
1285            ** History:
1286            **
1287            **    Date     Programmer             Modification
1288            **   --------   ----------   ---------------------------------
1289            **  09/14/82      FH        Designed and coded.
1290            **  02/24/83      FH        Expanded to 16 use levels
1291            **
```

```
1292                ****************************************************************
1293                ****************************************************************
1294 014DB 22      =R<RST2 P=     2              To save 3 levels.
1295 014DD D1      =R<RSTK B=0    A              Set carry for pointer increment
1296 014DF CD              B=B-1  A               .
1297 014E1 07              C=RSTK
1298 014E3 7D10            GOSUB  RSTK>1         Set up counters
1299 014E7 5E0             GONC   RSTK40         BET
1300                *
1301                *  Store stack level
1302                *
1303 014EA 164      RSTK20 D0=D0+ 5              Move position
1304 014ED 0C              P=P+1                  .
1305 014EF 560             GONC   RSTK40          . (continue unless wraparound)
1306 014F2 1902            D0=(2) =RSTKBF         .
1307 014F6 07       RSTK40 C=RSTK                Save level
1308 014F8 144             DAT0=C A               .
1309 014FB A4E             C=C-1  S              Decrement level count
1310 014FE 5BE             GONC   RSTK20          . and loop if more
1311 01501 45C             GOC    R>RS60         BET
1312
1313                ****************************************************************
1314                *
1315                *  RSTK>1  -  Set up counters for R<RSTK and RSTK<R.
1316                *
1317                *  Entry:  Carry  =  Set if buffer position to be incremented
1318                *                 =  Clear if buffer position already correct
1319                *          C(A)  =  Return address
1320                *          P     =  n - 1 (level count - 1)
1321                *
1322                *  Exit:   C(S)  =  n - 1
1323                *          B(A)  =  Return address
1324                *          P     =  Buffer pointer (incremented if needed)
1325                *          D0    @  Position in RSTKBF corresponding to P
1326                *          Carry  =  Clear
1327                *
1328 01504 D5       RSTK>1 B=C    A              Save return address
1329 01506 1BF1            D0=(5) =RSTKBp        Position D0 @ buffer pointer
           8F2
1330 0150D 1564            C=DAT0 S              Set P    = pointer,
1331 01511 80FF            CPEX   15              .  C(S) = n - 1
1332 01515 540             GONC   RSTK>2         No increment needed?
1333 01518 0C              P=P+1                 Increment
1334 0151A D2       RSTK>2 C=0    A              Make C(A) =     1*pointer
1335 0151C 80C0            C=P    0               .
1336 01520 C6              C=C+C  A               . (            2*pointer
1337 01522 C6              C=C+C  A               . (            4*pointer
1338 01524 809             C+P+1                  . (        1 + 5*pointer
1339 01527 132             ADOEX                 Calculate address =
1340 0152A CA              A=A+C  A               . RSTKBp + 1 + 5*pointer
1341 0152C 132             ADOEX                  .
1342 0152F 03              RTNCC
```

```
1343            SALLOC  STITLE Allocate Save Stack Area
1344            ********************************************************************
1345            ********************************************************************
1346            **
1347            ** Name:(S) SALLOC  -  Allocate Arbitrary Save Stack Block
1348            ** Name:    ALINFO  -  Allocate File Info Save Stack Block
1349            **
1350            ** Category:   SAVSTK
1351            **
1352            ** Purpose:
1353            **      Allocates a block of the specified size on the Save
1354            **      Stack (SAVSTK).  SALLOC allocates an arbitrary size,
1355            **      and ALINFO allocates the amount for the filespec info
1356            **      area used by COPY and TRANSFORM.  Available memory is
1357            **      checked with or without LEEWAY, depending on the entry
1358            **      conditions.
1359            **
1360            ** Entry:
1361            **      P  =  0 if memory check to be performed with LEEWAY
1362            **         #  0 if memory check to be performed without LEEWAY
1363            **     SALLOC:
1364            **      C(A)  =  Number of nibs to allocate
1365            **
1366            ** Exit:
1367            **      P      =  0
1368            **      B(A)   =  Number of nibs allocated
1369            **
1370            **     Carry clear:
1371            **      Allocation was successful
1372            **      AVMEME updated
1373            **      D1 @ Start of newly created Save Area
1374            **      C=D0 on entry.
1375            **
1376            **     Carry set:
1377            **      Allocation failed due to insufficient memory
1378            **      C(3-0) = Error code (=eMEM)
1379            **
1380            **
1381            ** Calls:      MEMCKL,MOVEU3
1382            **
1383            ** Uses.......
1384            **  Exclusive: A(A),B(A),C(A),D1
1385            **  Inclusive: A,   B(A),C(A),D1
1386            **
1387            ** Stk lvls:   2
1388            **
1389            ** Detail:   If sufficient memory to allocate
1390            **              Save D0 on stack
1391            **              Move memory between SAVSTK --> AVMEME
1392            **              Update AVMEME
1393            **              Restore D0
1394            **
1395            ********************************************************************
1396            ********************************************************************
1397            *
```

```
1398            * NOTE:  LC(5) NOT possible to preserve P setting on entry
1399            *
1400 01531 1F23 =ALINFO D1=(5) =1FILSV        Allocate 50 decimal nibs
          000
1401 01538 137          CD1EX                Move into C(A)
1402 0153B 766D =SALLOC GOSUB  MEMCKL        Insufficient memory?
1403 0153F 400          RTNC                 Error Return
1404 01542 136          CD0EX                Preserve D0
1405 01545 06           RSTK=C
1406            *
1407            * Move Memory between (SAVSTK) ---> (AVMEME)
1408            *
1409            * Compute Move Memory parameters
1410            *    Start of Source = Current AVMEME                (D0)
1411            *    Start of Dest   = (AVMEME) - Length to Alloc    (D1)
1412            *    Length to Move  = ((SAVSTK) - (AVMEME))         (C)
1413            * Update AVMEME
1414            *    D1 @ AVMEMS
1415            *    B = Length to Allocate
1416            *
1417 01547 174          D1=D1+ (AVMEME)-(AVMEMS)  Position to AVMEME
1418 0154A 143          A=DAT1 A             Current AVMEME
1419 0154D 1BE9         D0=(5) =SAVSTK
          5F2
1420 01554 146          C=DAT0 A
1421 01557 130          D0=A                 Start of Source = AVMEME
1422 0155A E2           C=C-A  A             Length to Move
1423 0155C E0           A=A-B  A             New AVMEME
1424 0155E 141          DAT1=A A             Update AVMEME
1425 01561 131          D1=A   A             Start of Destination
1426 01564 8E00         GOSUBL =MOveu3       Move memory
          00
1427 0156A 07           C=RSTK               Restore D0
1428 0156C 134          D0=C
1429 0156F 03           RTNCC                Successful Allocate
```

```
1430            SNAPRS  STITLE Restore CPU Snapshot
1431            ********************************************************************
1432            ********************************************************************
1433            **
1434            ** Name:(S) SNAPRS  -  Restore CPU Snapshot From SNAPSV Buffer
1435            ** Name:(S) SNAPR*  -  Restore CPU Snapshot From Any Buffer
1436            **
1437            ** Category:   SAVUTL
1438            **
1439            ** Purpose:
1440            **      Restore registers saved by SNAPSV (A, D, D0, D1) and
1441            **      return saved stack level for caller to push onto stack.
1442            **
1443            ** Entry:
1444            **     SNAPRS:
1445            **      None.
1446            **     SNAPR*:
1447            **      D1     @  Starting address of save buffer + 42 decimal
1448            **
1449            ** Exit:
1450            **      D0     =  Value saved by last SNAPSV call.
1451            **      D1     =  Value saved by last SNAPSV call.
1452            **      A      =  Value saved by last SNAPSV call.
1453            **      B(A)   =  Stack level saved by last SNAPSV call.
1454            **      C(A)   =  Stack level saved by last SNAPSV call.
1455            **      D      =  Value saved by last SNAPSV call.
1456            **      Carry  =  Clear.
1457            **
1458            ** Calls:      None.
1459            **
1460            ** Uses.......
1461            **  Inclusive: A, B(A), C(A), D, D0, D1
1462            **
1463            ** Stk lvls:   0
1464            **
1465            ** Detail:
1466            **
1467            **          SNAPSHOT SAVE BUFFER LAYOUT
1468            **
1469            **        Offset
1470            **        into
1471            **        Buffer      Nibs        Register
1472            **        ------      ----        --------
1473            **          0          16         A
1474            **         16          16         D
1475            **         32           5         D1
1476            **         37           5         D0
1477            **         42           5         Stack level
1478            **
1479            ** History:
1480            **
1481            **    Date     Programmer             Modification
1482            **  --------   ----------   ----------------------------------
1483            **  09/10/82      FH        Designed and coded.
1484            **
```

```
1485            ****************************************************************
1486            ****************************************************************
1487 01571 1FA1 =SNAPRS D1=(5) (=SNAPBF)+42   Fetch return address into B(A)
           8F2
1488 01578 143  =SNAPR* A=DAT1 A               .
1489 0157B D8          B=A    A               .
1490 0157D 1C4         D1=D1- 5               Restore D0
1491 01580 143         A=DAT1 A               .
1492 01583 130         D0=A                   .
1493 01586 1C4         D1=D1- 5               Fetch D1
1494 01589 147         C=DAT1 A               .
1495 0158C 1CF         D1=D1- 16              Restore D
1496 0158F AFF         CDEX   W               . (Do not pack this sequence out
1497 01592 1577        C=DAT1 W               .  since upper C(W) nibs must be
1498 01596 AFF         CDEX   W               .  perserved for msg routines)
1499 01599 1CF         D1=D1- 16              Restore A
1500 0159C 1537        A=DAT1 W               .
1501 015A0 135         D1=C                   Restore D1
1502 015A3 D9          C=B    A               Position stack level
1503 015A5 03          RTNCC
```

```
1504            BNAPSV  STITLE CPU Snapshot Save
1505            ************************************************************
1506            ************************************************************
1507            **
1508            ** Name:(S) SNAPSV  -  Save Snapshot of CPU in SNAPSV Buffer
1509            ** Name:    SNAPLC  -  Save Snapshot of CPU in Any Buffer
1510            **
1511            ** Category:   SAVUTL
1512            **
1513            ** Purpose:
1514            **      Save limited snapshot of CPU (1 stack level, A,D,D0,D1)
1515            **      to allow a routine to function without disturbing the
1516            **      registers of its caller.  Useful for tight situations.
1517            **      Snapshot is saved in system RAM, and is restored by the
1518            **      routine SNAPRS.
1519            **
1520            **      SNAPSV uses dedicated RAM locations for storage.
1521            **      SNAPLC uses a "local" RAM location for storage.
1522            **
1523            ** Entry:
1524            **    SNAPSV
1525            **      C(A)   =  Stack level to be saved; popped by caller of
1526            **                SNAPSV.
1527            **
1528            **    SNAPLC
1529            **      D1     @  Starting address of save buffer + 42 decimal
1530            **      C(A)   =  Anything you want to save.
1531            **
1532            ** Exit:
1533            **      B(A)   =  C(A) on entry
1534            **      C(A)   @  Save area start address + 42 decimal
1535            **      Carry  =  Clear.
1536            **
1537            **
1538            ** Calls:      None.
1539            **
1540            ** Uses.......
1541            **  Inclusive: B(A), C(A)
1542            **
1543            ** Stk lvls:   0
1544            **
1545            ** Detail:
1546            **
1547            **           SNAPSHOT SAVE BUFFER LAYOUT
1548            **
1549            **        Offset
1550            **        into
1551            **        Buffer      Nibs       Register
1552            **        ------      ----       --------
1553            **          0          16        A
1554            **         16          16        D
1555            **         32           5        D1
1556            **         37           5        D0
1557            **         42           5        Stack level
1558            **
```

```
1559            ** History:
1560            **
1561            **    Date      Programmer             Modification
1562            **  --------    ----------   ----------------------------------
1563            **  09/10/82       FH        Designed and coded.
1564            **  11/15/82       MB        Added SNAPLC entry
1565            **
1566            ********************************************************************
1567            ********************************************************************
1568 015A7 D5   =SNAPSV B=C    A               Set aside stack level
1569 015A9 137          CD1EX                  Save A
1570 015AC 1F0F         D1=(5) =SNAPBF          .
           7F2
1571 015B3 1517 =SNAPLC DAT1=A W               .
1572 015B7 17F          D1=D1+ 16              Save D
1573 015BA AFF          CDEX   W                .
1574 015BD 1557         DAT1=C W                .
1575 015C1 AFF          CDEX   W                .
1576 015C4 17F          D1=D1+ 16              Save D1
1577 015C7 145          DAT1=C A                .
1578 015CA 174          D1=D1+ 5               Save D0
1579 015CD 136          CD0EX                   .
1580 015D0 145          DAT1=C A                .
1581 015D3 136          CD0EX                   . and restore it
1582 015D6 DD           CBEX   A               Save stack level
1583 015D8 174          D1=D1+ 5                .
1584 015DB 145          DAT1=C A                .
1585 015DE DD           BCEX   A                . and return it to B(A)
1586 015E0 137          CD1EX                  Restore D1
1587 015E3 03           RTNCC
```

```
1588            SRLEAS  STITLE Release Block from Save Stack
1589            ****************************************************************
1590            ****************************************************************
1591            **
1592            ** Name:(S) SRLEAS  -  Release Arbitrary Block From Save Stack
1593            ** Name:    RLINFO  -  Release File Info Block From Save Stack
1594            **
1595            ** Category:   SAVUTL
1596            **
1597            ** Purpose:
1598            **      Release block of specified size from the Save Stack.
1599            **      SRLEAS releases a block of arbitrary size, while RLINFO
1600            **      releases a block the size of the filespec info area
1601            **      used by COPY and TRANSFORM.
1602            **
1603            ** Entry:
1604            **     SRLEAS:
1605            **      C(A)   =  Number of nibs to release.
1606            **     RLINFO:
1607            **      P      =  0
1608            **
1609            ** Exit:
1610            **      P      =  0
1611            **      D0     @  Old Av mem end
1612            **      D1     @  New Av mem end
1613            **      Carry  =  Clear
1614            **
1615            ** Calls:    MOVED3  (RLINFO falls into SRLEAS)
1616            **
1617            ** Uses.......
1618            **  Exclusive: A(A),B(A),C(A),D0,D1
1619            **  Inclusive: A,   B(A),C(A),D0,D1
1620            **
1621            **
1622            ** Stk lvls:   0
1623            **
1624            ** Detail:
1625            **          Move Memory Down parameters:
1626            **
1627            **          End Dest   = (SAVSTK)                        (D1)
1628            **          End Source = (SAVSTK) - release              (D0)
1629            **          Length     = ((SAVSTK) - release) - (AVMEME) (C)
1630            **
1631            ****************************************************************
1632            ****************************************************************
1633 015E5 3423 =RLINFO LC(5)  =1FILSV        Release 50 decimal nibs from stk
           000
1634 015EC 1F99 =SRLEAS D1=(5) =AVMEME
           5F2
1635 015F3 143          A=DAT1 A             Current Avail Mem End
1636 015F6 D8           B=A    A             Save it
1637 015F8 CA           A=A+C  A             New Avail Mem End
1638 015FA 141          DAT1=A A             Update AVMEME
1639 015FD 174          D1=D1+ (SAVSTK)-(AVMEME)   Position to SAVSTK ptr
1640 01600 143          A=DAT1 A             Save Stack
```

```
1641 01603 131          D1=A                    End of Destination
1642 01606 EE           C=A-C  A                End of Source
1643 01608 134          D0=C
1644 0160B E9           C=C-B  A                Length to Move
1645 0160D 8D00 =Moved3 GOVLNG =MOVED3          Move stack & RTNCC
           000
```

```
1646                    UPDFCL  STITLE Update File Chain Length
1647                    ********************************************************************
1648                    ********************************************************************
1649                    **
1650                    ** Name:    UPDFCL  -  Update File Chain Length
1651                    ** Name:    UPDFC+  -  Update File Chain Length
1652                    **
1653                    ** Category:   FILUTL
1654                    **
1655                    ** Purpose:
1656                    **      Add an update offset to the file chain length field of
1657                    **      a file header.
1658                    **
1659                    ** Entry:
1660                    **      B(A)    =  Update offset to add to header
1661                    **     UPDFCL:
1662                    **      C(A)    =  Address of file header
1663                    **     UPDFC+:
1664                    **      D1      @  File header
1665                    **
1666                    ** Exit:
1667                    **      B(A)    =  Entry state
1668                    **      C(A)    =  New file chain length
1669                    **      D1      @  File chain length field of file header
1670                    **      Carry   =  Clear
1671                    **
1672                    ** Calls:      None.
1673                    **
1674                    ** Uses.......
1675                    **  Inclusive: B(A), C(A), D1
1676                    **
1677                    ** Stk lvls:   0
1678                    **
1679                    **
1680                    ********************************************************************
1681                    ********************************************************************
1682 01614 135        =UPDFCL D1=C
1683 01617 17F        =UPDFC+ D1=D1+ (oFTYPh)
1684 0161A 17F                D1=D1+ (oFLENh)-(oFTYPh)
1685 0161D 147                C=DAT1 A
1686 01620 C9                 C=C+B  A
1687 01622 145                DAT1=C A
1688 01625 03                 RTNCC
1689                    *
1690                    *  Define the END length for packing info
1691                    *
1692 01627              END
```

```
 ADJR20  Abs    5062 #013C6 -   815   813
=ADJREF  Abs    5033 #013A9 -   807
=ALINFO  Abs    5425 #01531 -  1400
 AVMEME  Abs  193945 #2F599 -    14   384   486  1417  1634  1639
 AVMEMS  Abs  193940 #2F594 -    14   386   491  1122  1127  1131  1152  1417
=AVS=C   Abs    5271 #01497 -  1152
=AVS=DO  Abs    5268 #01494 -  1151
=BBCOLL  Abs    5218 #01462 -  1134
=CHKSPC  Abs    4782 #012AE -   481
=CHKSPF  Abs    4802 #012C2 -   487
=CHKmem  Abs    4807 #012C7 -   489
 CKSPC1  Abs    4795 #012BB -   486   480
 CKSPC2  Abs    4827 #012DB -   495   485   490
=CLCOLL  Abs    5192 #01448 -  1126
 CLCSTK  Abs  193925 #2F585 -    14  1126  1127
 CSLW5   Ext                 -   174
 CSRW5   Ext                 -   187
 CURDVC  Ext                 -    56
 CVUCW   Ext                 -   316
 D1=CRS  Ext                 -    58
=D1=IBS  Abs    5234 #01472 -  1139
=D1@AVS  Abs    4761 #01299 -   386
=DEFFIL  Abs    4401 #01131 -    56
 END     Abs    5671 #01627 -  1692
 EOFLC+  Ext                 -   379
 FIL010  Abs    4588 #011EC -   278
 FIL020  Abs    4621 #0120D -   289   280   282   285
 FIL030  Abs    4639 #0121F -   295   291
 FIL035  Abs    4652 #0122C -   299   296
 FIL040  Abs    4666 #0123A -   304   302
 FIL045  Abs    4689 #01251 -   312   304
 FIL050  Abs    4706 #01262 -   315   297   298
=FILFIL  Abs    4558 #011CE -   269
 FLAD20  Abs    4751 #0128F -   384   375
 FLAD30  Abs    4758 #01296 -   385  1145
=FLADDR  Abs    4715 #0126B -   374   736
 FLDE20  Abs    4457 #01169 -   158   156
 FLDE40  Abs    4460 #0116C -   159   164
 FLDE50  Abs    4462 #0116E -   160   162
 FLDE60  Abs    4464 #01170 -   161   153
 FLDE80  Abs    4493 #0118D -   172   168
 FLDE85  Abs    4524 #011AC -   184   182
=FLDE90  Abs    4550 #011C6 -   193   170
=FLDEV+  Abs    4433 #01151 -   149
=FLDEVX  Abs    4436 #01154 -   150    57
 INBS    Abs  194246 #2F6C6 -    14  1135  1139
=INBS=C  Abs    5222 #01466 -  1135
 INITPT  Ext                 -  1124
 LEEWAY  Abs     212 #000D4 -    13   483   877
 LOCADR  Ext                 -   717
 LSTADR  Ext                 -   381
=MEMCKL  Abs    4773 #012A5 -   477  1402
=MEMCL+  Abs    4775 #012A7 -   478
 MFD010  Abs    4859 #012FB -   566   561   564
 MFD020  Abs    4870 #01306 -   570   568   734
```

```
=MFDEVC  Abs    4837 #012E5 -   558   272
=MFDVC+  Abs    4831 #012DF -   557
=MFDVC-  Abs    4840 #012E8 -   559   289
=MOVE*M  Abs    4872 #01308 -   607   759   966
 MOVE10  Abs    4898 #01322 -   616   609
 MOVED3  Ext         #      - 1645
 MOVEDM  Ext         #      -   612
 MOVEUM  Ext         #      -   616
 MOveu3  Ext         #      - 1426
 MVME00  Abs    4917 #01335 -   710   722
 MVME20  Abs    4955 #0135B -   723   720
 MVME40  Abs    4989 #0137D -   736   732
 MVME60  Abs    5005 #0138D -   744   738
 MVME80  Abs    5020 #0139C -   755   747
=MVMEM   Abs    4963 #01363 -   726
=MVMEM+  Abs    4924 #0133C -   713   956
=Moved3  Abs    5645 #0160D - 1645
 OBCOL1  Abs    5182 #0143E - 1123  1128  1132
=OBCOLL  Abs    5173 #01435 - 1121
=OBLCMP  Abs    5260 #0148C - 1147   950
=OBPRD   Abs    5249 #01481 - 1144  1147
 OUTBS   Abs  193935 #2F58F -   14   964  1121  1122  1144
 PRGFE   Ext         #      -   710
=R<RST2  Abs    5339 #014DB - 1294
=R<RSTK  Abs    5341 #014DD - 1295
 R>RS20  Abs    5296 #014B0 - 1219  1226
 R>RS40  Abs    5313 #014C1 - 1225  1223
 R>RS60  Abs    5319 #014C7 - 1230  1311
 RDINFD  Ext         #      -   278
 RDINFO  Ext         #      -   557
 RDINFS  Ext         #      -   271
 RFAD+I  Ext         #      -   814
 RFAD-I  Ext         #      -   815
=RLINFO  Abs    5605 #015E5 - 1633
 RMEM05  Abs    5101 #013ED -   884   871
=RMEM10  Abs    5103 #013EF -   885   495   869   880   882
=RMEMCH  Abs    5069 #013CD -   867   739
 ROMF-   Ext         #      -   177
=RPLLIN  Abs    5111 #013F7 -   948
=RST2<R  Abs    5286 #014A6 - 1212
 RSTK20  Abs    5354 #014EA - 1303  1310
 RSTK40  Abs    5366 #014F6 - 1307  1299  1305
=RSTK<R  Abs    5288 #014A8 - 1213
 RSTK>1  Abs    5380 #01504 - 1328  1215  1298
 RSTK>2  Abs    5402 #0151A - 1334  1332
 RSTKBF  Abs  194592 #2F820 -   14  1224  1306
 RSTKBp  Abs  194591 #2F81F -   14  1231  1329
=SALLOC  Abs    5435 #0153B - 1402
 SAVSTK  Abs  193950 #2F59E -   14  1419  1639
 SCRTCH  Abs  194817 #2F901 -   14   809
 SNAPBF  Abs  194544 #2F7F0 -   14  1487  1570
=SNAPLC  Abs    5555 #015B3 - 1571
=SNAPR*  Abs    5496 #01578 - 1488
=SNAPRS  Abs    5489 #01571 - 1487
=SNAPSV  Abs    5543 #015A7 - 1568
```

```
=SRLERS  Abs    5612 #015EC -  1634
 SVINFO  Ext                 -   314
=SYCOLL  Abs    5205 #01455 -  1130  1134
 SYSEN   Abs  193930 #2F58A -    14  1130  1131
=UPDFC+  Abs    5655 #01617 -  1683
=UPDFCL  Abs    5652 #01614 -  1682   751
 dCARD   Abs       7 #00007 -    13   566
 eDVCNF  Ext                 -   193
 eMEM    Ext                 -   885
=initpt  Abs    5185 #01441 -  1124
 lFILSV  Abs      50 #00032 -    13  1400  1633
 oFLENh  Abs      32 #00020 -    13  1684
 oFTYPh  Abs      16 #00010 -    13  1683  1684
 sCARD   Abs       2 #00002 -    13   281   558   569
 sDEST   Abs       3 #00003 -    13   155   169   305
 sEXTDV  Abs       0 #00000 -    13   269   565
 sUNDEF  Abs       1 #00001 -    13   270   303
 svinfo  Abs    4700 #0125C -   314
```

Input Parameters

  Source file name is TI&UTL::MS

  Listing file name is TI/UTL:TI:ML::-1

  Object file name is TI%UTL:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                      TITLE  Command Stack Utilities<831212.1206>
 2 01627                ABS    #1627
 3              *     SSS   BBBB    &      CCC   M   M  DDDD
 4              *    S   S  B   B  & &    C   C  MM MM   D  D
 5              *    S      B   B  & &    C      M M M   D  D
 6              *     SSS   BBBB    &     C      M M M   D  D
 7              *        S  B   B  & & &  C      M   M   D  D
 8              *    S   S  B   B  &  &   C   C  M   M   D  D
 9              *     SSS   BBBB    && &   CCC   M   M  DDDD
10
11                      RDSYMB SB%RAM::MS
12             ********************************************************************
13             ********************************************************************
14             **
15             ** Name:(S) CMDPR"  -  Text for command stack prompt
16             **
17             ** Category:   DSPUTL
18             **
19             ** Purpose:
20             **      This is the text for the command stack prompt, it is
21             **      the following sequence:  CR, LF, cursor off,
22             **      backslash, cursor on.  The text string is terminated
23             **      by a FF byte as expected by BF2DSP.
24             **
25             ** Entry:
26             **      Don't enter
27             **
28             ** Exit:
29             **
30             ** History:
31             **
32             **    Date     Programmer             Modification
33             **  --------   ----------   ----------------------------------
34             **  11/09/83   B.S.         Added documentation
35             **
36             ********************************************************************
37             ********************************************************************
38 01627 D0A0 =CMDPR" NIBHEX D0A0           CR/LF
39 0162B B1C3         NIBHEX B1C3           Cursor off
40 0162F B125         NIBHEX B125           Replace cursor
41 01633 C5           NIBHEX C5             Backslash prompt
42 01635 B1E3         NIBHEX B1E3           Cursor on
43 01639 FF           NIBHEX FF
44             *-
45             *-
```

```
46                      EJECT
47                ********************************************************************
48                ********************************************************************
49                **
50                ** Name:    CMDSTK  -  CMDS key
51                **
52                ** Category:   KEYUTL
53                **
54                ** Purpose:
55                **      Process CMDS key.  Toggles command stack mode and puts
56                **      up command stack line if necessary.
57                **
58                ** Entry:
59                **      P       =  0
60                **
61                ** Exit:
62                **      Exits through CMDS00
63                **
64                ** Calls:       SFLAGT,BF2DSP,CMDS00
65                **
66                ** Uses:
67                **  Exclusive:  C(B),D1
68                **  Inclusive:  A,B,C,D,D0,D1
69                **
70                ** Stk lvls:   5
71                **
72                ** History:
73                **
74                **    Date      Programmer              Modification
75                **  --------    ----------    ----------------------------------
76                **  07/28/83    B.S.          Added documentation
77                **
78                ********************************************************************
79                ********************************************************************
80 0163B 3100 =CMDSTK LC(2)   =f1CMDS
81 0163F 8F00         GOSBVL  =SFLAGT         Toggle Command Stack Flag
         000
82 01646 5C1          GONC    CMDS00
83 01649 1F00         D1=(5)  =CRLFPR         CR/LF ">"
         000
84 01650 6000 =Bf2Dsp GOTO    =BF2DSP
85                *-
86                *-
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) CMD1ST  -  Set command stack pointer to 1st cmd
                **
                ** Category:   KEYUTL
                **
                ** Entry:
                **      None
                **
                ** Exit:
                **      D1 points to CMDPTR
                **      C(A)=0
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Exclusive: C(A)
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   ----------------------------------
                **  07/28/83   B.S.         Added documentation
                **
                ****************************************************************
                ****************************************************************
01654 D2        =CMD1ST C=0     A
01656 1F4D              D1=(5) =CMDPTR        Set pointer to first command
      6F2
0165D 15D0              DAT1=C 1
01661 03                RTNCC
                *-
                *-
```

```
123                     EJECT
124                 ****************************************************************
125                 ****************************************************************
126                 **
127                 ** Name:     CMDS00  -  Display Cmd Stack Entry
128                 ** Name:     CMDS10  -  Display Cmd Stack Entry
129                 ** Name:(S) CMDS20  -  Display Cmd Stack Entry
130                 **
131                 ** Category:   KEYUTL
132                 **
133                 ** Purpose:
134                 **      CMDS00 - Initializes to first command stack entry then
135                 **      CMDS10 - Puts up command stack prompt then
136                 **      CMDS20 - Puts up command stack entry and moves cursor
137                 **               to far left.
138                 ** Entry:
139                 **      P      =  0
140                 **      CMDS10 and CMDS20 require that CMDPTR be set to specify
141                 **           which command should be displayed.
142                 **
143                 ** Exit:
144                 **      P      =  0
145                 **
146                 ** Calls:      BF2DSP,CMDFND,DSPCNA,CURSFL,CMD1ST
147                 **
148                 ** Uses.......
149                 **  Exclusive: D1,C(A),A(W)
150                 **  Inclusive: D0,D1,A,B,C,D
151                 **
152                 ** Stk lvls:   5
153                 **
154                 ** History:
155                 **
156                 **    Date     Programmer             Modification
157                 **  --------   ----------   ---------------------------------
158                 **  07/28/83   B.S.         Added documentation
159                 **
160                 ****************************************************************
161                 ****************************************************************
162 01663 7DEF  CMDS00 GOSUB  CMD1ST
163 01667 1F72 =CMDS10 D1=(5) =CMDPR"
          610
164 0166E 7000         GOSUB  =BF2DSP        Clear display and put out prompt
165 01672 7D10 =CMDS20 GOSUB  CMDFND
166 01676 172          D1=D1+ 3              D1 points to start of text
167 01679 C2           C=C+A  A              Calculate end of this buffer
168 0167B 81C          ASRB                 Divide by two(throws away odd lowes
169 0167E CC           A=A-1  A              Subtract 2 nibbles from length
170 01680 CC           A=A-1  A              Don't output CR.
171 01682 490          GOC    cursfl         If buffer length zero then skip
172 01685 8F00         GOSBVL =DSPCNA        Send buffer to display
          000
173 0168C 8D00 =cursfl GOVLNG =CURSFL        Move cursor to far left of display
          000
174                 *-
```

```
    175             *-
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) CMDFND  -  Find Nth Command Stack Entry
                **
                ** Category:   KEYUTL
                **
                ** Purpose:
                **      Finds the command stack entry indicated by CMDPTR
                **
                ** Entry:
                **      CMDPTR is number of entry to find (0-->first,F-->15th)
                **
                ** Exit:
                **      D1 points to start of cmd stack entry (at length field)
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: D1,A(W),C(A)
                **
                ** Stk lvls:   0
                **
                ** Detail:
                **      This routine starts with the newest command (pointed to
                **      by RAWBFR) and chains up stack toward the oldest entry
                **      until the specified entry is reached.
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------   ----------------------------------
                **  07/28/83   B.S.         Added documentation
                **
                ****************************************************************
                ****************************************************************
01693 1F4D 6F2  =CMDFND D1=(5) =CMDPTR
0169A AF0       A=0     W
0169D 1534      A=DAT1  S               Read stack pointer
016A1 1E08 5F   D1=(4)  =RAWBFR
016A7 147       C=DAT1  A
016AA 135       D1=C                    D1=(RAWBFR)
016AD 1C2    CMDF10 D1=D1- 3            Back up 3 to start of length
016B0 1533      A=DAT1  X               Read length
016B4 137       CD1EX
016B7 E2        C=C-A   A               Subtract length to get start
016B9 135       D1=C                    Point D1 to start of buffer(length)
016BC A4C       A=A-1   S               Decrement loop counter
016BF 5DE       GONC    CMDF10          Done? No, then loop back for next c
016C2 03        RTNCC
                *-
                *-
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) CMDINI  -  Recalls CMDPTR and MAXCMD
                **
                ** Category:   KEYUTL
                **
                ** Purpose:
                **      Recall CMDPTR and MAXCMD to A(0) and C(0)
                **
                ** Entry:
                **      None
                **
                ** Exit:
                **      A(0)   =  (CMDPTR)
                **      C(0)   =  (MAXCMD)
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: D1,C(0),A(0)
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------   ---------------------------------
                **  07/28/83   B.S.         Added documentation
                **
                ********************************************************************
                ********************************************************************
016C4 3100      =CMDST? LC(2)  =flCMDS
016C8 8E00              GOSUBL =sflag?
      00
016CE 500               RTNNC
016D1 1F67      =CMDINI D1=(5) =MAXCMD
      9F2
016D8 15F0              C=DAT1 1
016DC 1E4D              D1=(4) =CMDPTR
      6F
016E2 15B0              A=DAT1 1
016E6 01                RTN
                *-
                *-
                ********************************************************************
                ********************************************************************
                **
                ** Name:     CURSUj  -  Cmd Stack Keys
                ** Name:     CURSDj  -  Cmd Stack Keys
                ** Name:     CURSTj  -  Cmd Stack Keys
                ** Name:     CURSBj  -  Cmd Stack Keys
                **
                ** Category:   SYSTEM
```

```
                    **
                    ** Purpose:
                    **      These entry points handle the cursor up, down, top and
                    **      bottom keys.  It checks if in command stack mode and
                    **      either recalls a program line or a commands stack line.
                    **
                    ** Entry:
                    **      P       =  0
                    **
                    ** Exit:  Jumps to MAIN10 when line has been recalled
                    **
                    ** Calls:       CMDST?,CMDS10
                    **
                    ** Stk lvls:    1
                    **
                    ** History:
                    **
                    **    Date      Programmer              Modification
                    **  --------    ----------   ----------------------------------
                    **  07/28/83    B.S.         Added documentation
                    **
                    ********************************************************************
                    ********************************************************************
016E8 78DF =CURSUp  GOSUB   CMDST?
016EC 490           GOC     CMDSTU
016EF 8D00          GOVLNG  =CURSRU
      000
                    *-
                    *-
016F6 B04   CMDSTU  A=A+1   P               Move up one stack item
016F9 4B0           GOC     CMDSTX          Watch for overflow(NOP)
016FC 986           ?A>C    P               Is it greater than maxcmd?
016FF 60            GOYES   CMDSTX          Yes, then don't change pointer.
01701 1590  CMDSTN  DAT1=A  1               Update cmd stack ptr with New value
01705 7E5F  CMDSTX  GOSUB   CMDS10
01709 8C00          GOLONG  =MAIN10
      00
                    *-
                    *-
0170F 71BF =CURSDp  GOSUB   CMDST?
01713 490           GOC     CMDSTD
01716 8D00          GOVLNG  =CURSRD
      000
                    *-
                    *-
0171D A0C   CMDSTD  A=A-1   P               Move down one stack item
01720 44E           GOC     CMDSTX          Watch for underflow(NOP)
01723 5DD           GONC    CMDSTN          (B.E.T.) Change pointer
                    *-
                    *-
01726 7A9F =CURSTp  GOSUB   CMDST?
0172A 490           GOC     CMDSTT
0172D 8D00          GOVLNG  =CURTOP
      000
                    *-
```

```
    332             *-
    333 01734 1550   CMDSTT DAT1=C P              Copy (MAXCMD) into (CMDPTR)
    334 01738 6CCF          GOTO   CMDSTX
    335             *-
    336             *-
    337 0173C 748F  =CURSBj GOSUB  CMDST?
    338 01740 490           GOC    CMDSTB
    339 01743 8D00          GOVLNG =CURBOT
              000
    340             *-
    341             *-
    342 0174A AE0    CMDSTB A=0    B
    343 0174D 63BF          GOTO   CMDSTN
```

```
344                 STITLE MAKEBF  -  Make Buffer
345               ****************************************************************
346               ****************************************************************
347               **
348               ** Name:(S) MAKEBF  -  Make ASCII Buffer from Display Buffer
349               **
350               ** Category:   DSPUTL
351               **
352               ** Purpose:
353               **      Builds an ASCII buffer containing all readable
354               **      characters in the display and appends it to the
355               **      command stack (between CLCBFR and RAWBFR).
356               **
357               ** Entry:
358               **
359               ** Exit:
360               **      P      =  0
361               **      C(A) points at first char of text
362               **      D0 points past text
363               **      A(A)=Buffer length + 3 nibbles
364               **
365               ** Calls:      OUT1Tk,OUTBYT,OUTNBC,INITPT,STKCMD,CHKSPC,MOVEU2,
366               **             D=AVME
367               **
368               ** Uses.......
369               **  Exclusive: D0,D1,A,B,C,D(A)
370               **  Inclusive: D0,D1,A,B,C,D(A)
371               **
372               ** Stk lvls:   2
373               **
374               ** Detail:
375               **      D0 is initialized to contents of RAWBFR, a 3 nibble
376               **      length field is output, then for each readable
377               **      character in display, a byte is added to the buffer
378               **      by calling OUT1TK.  After buffer is built, a CR is
379               **      written to the end of the buffer.  STKCMD is called
380               **      to edit the command stack.  Pointers from RFNBFR
381               **      to AVMEMS are updated to point to new end of buffer.
382               **      If there is less than LEEWAY memory left, commands
383               **      in the command are crushed, starting with the oldest,
384               **      until LEEWAY available memory exists or all but the
385               **      most recent command have been crushed.
386               **
387               ** History:
388               **
389               **    Date     Programmer             Modification
390               **  --------   ----------   -----------------------------------
391               **  10/19/82   B.S.         Updated documentation
392               **
393               ****************************************************************
394               ****************************************************************
```

```
                        EJECT
                *****************************************************************
                *****************************************************************
                **
                ** Name:    GETLEE  -  Get LEEWAY by crushing command stack
                **
                ** Category:   GENUTL
                **
                ** Purpose:
                **      Deletes items from command stack until at least LEEWAY
                **      memory is available.
                **
                ** Entry:
                **      C(A)   =  (RAWBFR)
                **
                ** Exit:
                **      P      =  0
                **
                ** Calls:      CKSPC,MOVEU2
                **
                ** Uses.......
                **  Inclusive: A(W),C(W),D0,D1,P,
                **
                ** Stk lvls:   1
                **
                ** Algorithm:
                **    While there are uncrushed commands in stack and
                **         there is less than leeway left do
                **      Crush a command
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ----------------------------------
                **  10/25/83   B.S.         Added documentation
                **
                *****************************************************************
                *****************************************************************
01751 1B08 5F2  =MAKEBF D0=(5) =RAWBFR
01758 146       C=DAT0 A              Read in ptr
0175B 134       D0=C                  Store ptr in D0
0175E 8F00 000  GOSBVL =D=AVME        D(A)=(AVMEME)
01765 D2        C=0    A
01767 22        P=     2
01769 8E00 00   GOSUBL =OUTNBC        Output length
0176F 1FC4 5F2  D1=(5) (=DSPMSK)+12   Point to dsp mask(first 48 bits)
01776 1575      C=DAT1 M              Read in 48 bits
0177A 1E08 4F   D1=(4) =DSPBFS
01780 7ED0      GOSUB  MKBF20
01784 461       GOC    MKBF10
```

```
445 01787 1E04          D1=(4) =DSPMSK        Point to dsp mask(second 48 bits)
          5F
446 0178D 1575          C=DAT1 M              Read in 48 bits
447 01791 1E0E          D1=(4) (=DSPBFS)+2*48 Point to second half of display
          4F
448 01797 77C0          GOSUB  MKBF20
449 0179B 31D0  MKBF10  LCHEX  0D
450 0179F 8E00          GOSUBL =OUTBYT        Output CR and return
          00
451 017A5 136           CDOEX
452 017A8 1B08          D0=(5) =RAWBFR
          5F2
453 017AF 142           A=DAT0 A
454 017B2 132           ADOEX
455 017B5 164           D0=D0+ 5              Skip len and CR
456 017B8 132           ADOEX
457 017BB 8A2           ?A=C   A              Is it a null buffer?
458 017BE 83            GOYES  MKBF50         Yes, don't put in command stack
459 017C0 144           DAT0=C A
460 017C3 8F00          GOSBVL =STKCMD        Push line into command stack
          000
461 017CA 1F08  MKBF12  D1=(5) =RAWBFR
          5F2
462             NPtr    EQU    ((AVMEMS)-(RAWBFR))/5+1
463 017D1 2B            P=     16-(NPtr)
464 017D3 7000          GOSUB  =initpt        Init mem pointers
465 017D7 134  =GETLEE  D0=C
466 017DA D2            C=0    A
467 017DC 7000          GOSUB  =CHKSPC        Check if leeway available
468 017E0 4B2           GOC    MKBF16         No, then collapse a stack item
469 017E3 136   MKBF14  CDOEX
470 017E6 134           D0=C
471 017E9 D0            A=0    A
472 017EB 182           D0=D0- 3              Point at length field
473 017EE 1523          A=DAT0 X
474 017F2 E2            C=C-A  A              C(A) points at text
475 017F4 03            RTNCC                 Return from MAKEBF
476             *-
477             *-
478 017F6 D2    MKBF50  C=0    A
479 017F8 305           LC(1)  5
480 017FB DA            A=C    A              A(A)=Text len+3 nibbles
481 017FD 34B8          LC(5)  =NULLBF        C(A) points to start of text
          810
482 01804 134           D0=C
483 01807 161           D0=D0+ 2              D0 points past text
484 0180A 03            RTNCC
485             *-
486             *-
487             *   D0 points past nth command (end of source)
488 0180C 1F67  MKBF16  D1=(5) =MAXCMD
          9F2
489 01813 1574          C=DAT1 S
490 01817 1E67          D1=(4) =CLCBFR
          5F
```

```
491 0181D 147          C=DAT1 A
492 01820 135          D1=C                 D1 points at oldest buffer
493 01823 1C5          D1=D1- 6             Preadjust pointer
494 01826 D0           A=0    A
495 01828 A4E  MKBF18  C=C-1  S             Are all buffers collapsed?
496 0182B 47B          GOC    MKBF14        Yes, then continue without leeway
497 0182E 175          D1=D1+ 6
498 01831 1533         A=DAT1 X
499 01835 938          ?A=0   X             Is this buffer zero length?
500 01838 0F           GOYES  MKBF18        Yes, then look again for another
501 0183A 3500         LCHEX  003000
          0300
502 01842 15D5         DAT1=C 6             Make this buffer zero length
503 01846 175          D1=D1+ 6             D1 = start of destination
504 01849 137          CD1EX
505 0184C 135          D1=C                 Copy this to C(A)
506 0184F CA           A=A+C  A             A(A) = start of source
507 01851 136          CD0EX
508 01854 8F00         GOSBVL =MOVEU2       Move memory to fill gap
          000
509 0185B 137          CD1EX                C(A)=New AVMEMS(etc)
510 0185E 6B6F         GOTO   MKBF12        Update pointers and test if enough
511                                         memory is available yet
512             *-
513             *-
514 01862 31F2 MKBF20  LC(2)  47            Repeat 48 times
515 01866 AF5          B=C    W             Bit mask to B(M), Count to B(B)
516 01869 148  MKBF30  A=DAT1 B             Read byte
517 0186C 171          D1=D1+ 2             Move pointer to next byte
518 0186F 968          ?A=0   B             Is it a null byte?
519 01872 00           RTNYES               Yes, then finish up
520 01874 A55          B=B+B  M             Check for readable/unreadable
521 01877 480          GOC    MKBF40        Skip output if unreadable
522 0187A 8E00         GOSUBL =OUT1TK       Copy char to output buffer
          00
523 01880 A6D  MKBF40  B=B-1  B             Decrement char count
524 01883 55E          GONC   MKBF30        Loop back until done
525 01886 03           RTNCC                Done normally, return carry clear
526             *-
527             *-
528 01888 200          CON(3) 2             Buffer length is 2
529 0188B D0   =NULLBF NIBHEX D0            Carriage return
530 0188D 500          CON(3) 5             Buffer length is 2 (+ 3 for len)
```

```
 AVMEMS  Abs  193940 #2F594 -    11   462
 BF2DSP  Ext                -    84   164
=Bf2Dsp  Abs    5712 #01650 -    84
 CHKSPC  Ext                -   467
 CLCBFR  Abs  193910 #2F576 -    11   490
=CMD1ST  Abs    5716 #01654 -   117   162
 CMDF10  Abs    5805 #016AD -   219   225
=CMDFND  Abs    5779 #01693 -   213   165
=CMDINI  Abs    5841 #016D1 -   265
=CMDPR"  Abs    5671 #01627 -    38   163
 CMDPTR  Abs  194260 #2F6D4 -    11   118   213   267
 CMDS00  Abs    5731 #01663 -   162    82
=CMDS10  Abs    5735 #01667 -   163   314
=CMDS20  Abs    5746 #01672 -   165
=CMDST?  Abs    5828 #016C4 -   262   304   318   328   337
 CMDSTB  Abs    5962 #0174A -   342   338
 CMDSTD  Abs    5917 #0171D -   323   319
=CMDSTK  Abs    5691 #0163B -    80
 CMDSTN  Abs    5889 #01701 -   313   325   343
 CMDSTT  Abs    5940 #01734 -   333   329
 CMDSTU  Abs    5878 #016F6 -   309   305
 CMDSTX  Abs    5893 #01705 -   314   310   312   324   334
 CRLFPR  Ext                -    83
 CURBOT  Ext                -   339
=CURSBj  Abs    5948 #0173C -   337
=CURSDj  Abs    5903 #0170F -   318
 CURSFL  Ext                -   173
 CURSRD  Ext                -   320
 CURSRU  Ext                -   306
=CURSTj  Abs    5926 #01726 -   328
=CURSUj  Abs    5864 #016E8 -   304
 CURTOP  Ext                -   330
 D=AVME  Ext                -   436
 DSPBFS  Abs  193664 #2F480 -    11   442   447
 DSPCNA  Ext                -   172
 DSPMSK  Abs  193856 #2F540 -    11   440   445
=GETLEE  Abs    6103 #017D7 -   465
 MAIN10  Ext                -   315
=MAKEBF  Abs    5969 #01751 -   433
 MAXCMD  Abs  194934 #2F976 -    11   265   488
 MKBF10  Abs    6043 #0179B -   449   444
 MKBF12  Abs    6090 #017CA -   461   510
 MKBF14  Abs    6115 #017E3 -   469   496
 MKBF16  Abs    6156 #0180C -   488   468
 MKBF18  Abs    6184 #01828 -   495   500
 MKBF20  Abs    6242 #01862 -   514   443   448
 MKBF30  Abs    6249 #01869 -   516   524
 MKBF40  Abs    6272 #01880 -   523   521
 MKBF50  Abs    6134 #017F6 -   478   458
 MOVEU2  Ext                -   508
 NPtr    Abs       5 #00005 -   462   463
=NULLBF  Abs    6283 #0188B -   529   481
 OUT1TK  Ext                -   522
 OUTBYT  Ext                -   450
 OUTNBC  Ext                -   439
```

RAWBFR	Abs	193920	#2F580	-	11	462	216	433	452	461
SFLAGT	Ext			-	81					
STKCMD	Ext			-	460					
=cursfl	Abs	5772	#0168C	-	173	171				
flCMDS	Ext			-	80	262				
initpt	Ext			-	464					
sflag?	Ext			-	263					

Input Parameters

  Source file name is SB&CMD::MS

  Listing file name is SB/CMD:TI:ML::-1

  Object file name is SB%CMD:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS   BBBB    &     DDDD    SSS   PPPP
 2            *    S   S  B   B  & &     D  D  S   S  P   P
 3            *    S      B   B  & &     D  D  S      P   P
 4            *     SSS   BBBB    &      D  D   SSS   PPPP
 5            *        S  B   B  & & &   D  D      S  P
 6            *    S   S  B   B  &  &    D  D  S   S  P
 7            *     SSS   BBBB    && &  DDDD    SSS   P
 8            *
 9                     TITLE  Display Driver <831216.1505>
10 01890              ABS    #1890
11            *
12            *Annunciator bit definitions
13            *
14             LeftA  EQU    3
15             RightA EQU    4
16            *
17            *
18            *Status bit definitions
19            *
20             RArrow EQU    0
21             CurLft EQU    0
22             AtEnd  EQU    0
23
24            =BitsOK EQU    1
25             Phase  EQU    2
26             NoCur  EQU    3
27             UpdOff EQU    4
28            =Clear  EQU    5
29            =CurOff EQU    6
30            =Insert EQU    7
31            =ResetC EQU    8
32            =XDelay EQU    9
33             TimOut EQU    10
34             NoChFC EQU    11
35            *
36            * Time system timeout bit
37            *
38             fTMOUT EQU    3
39            *
40            *Escape status definitions
41            *
42             EscSt0 EQU    0              Not in esc seq
43             EscSt1 EQU    1              Esc has been received
44             EscSt2 EQU    2              Esc % has been received
45             EscSt3 EQU    3              Esc % <Row> has been received
46            *
47            *Stack levels to save in DSPCHA
48            *
49            =SavLvl EQU    5              Save 6 levels
50                     RDSYMB SB%RAM::MS
```

```
                    STITLE BLDDSP
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) BLDDSP  -  Build Display Pattern from Buffer
                ** Name:(S) BLDLCD  -  BLDDSP Except Display Status Active
                **
                ** Category:   DSPUTL
                **
                ** Purpose:
                **      Uses the display buffer and related status information
                **      to build the display bit pattern.
                **
                ** Entry:
                **      Hexmode
                **
                ** Exit:
                **      P      =  0
                **      Hexmode
                **
                ** Calls:      GETSTA,D0=FC,fCALC?,D10=FC,BLDBIT,BLDB40,
                **             WRTTM1,SETSTA
                **
                ** Uses.......
                **  Inclusive: A(W),B(W),C(W),D(W),D0,D1
                **
                ** Stk lvls:   2
                **
                ** Algorithm:
                **      If cursor is on then adjust FIRSTC so that cursor will
                **         be in display window.
                **      Turns left arrow annunciator on or off depending on
                **         whether FIRSTC is zero or not.
                **      Sets up registers and calls BLDBIT to build display.
                **      Turns on right arrow annunciator iff display buffer
                **         contains characters to the right of last character
                **         in the currently displayed window.
                **      If cursor is on then sets the cursor phase so the
                **         cursor will appear "on" first and falls into
                **         code for display update (ie cursor blink).
                **      If cursor off then disables display timer and returns.
                **
                ** History:
                **
                **    Date     Programmer            Modification
                **  --------   ----------   ----------------------------------
                **  10/19/82   B.S.         Updated documentation
                **
                ****************************************************************
                ****************************************************************
                *-
01890 67E2      PUTSTj GOTO   PUTSTA
                *-
                *-
01894 7673      =BF2BLD GOSUB  BF2DSP        Send buffer to display
```

```
    106 01898 77F2 =BLDDSP GOSUB  GETSTA
    107 0189C 871  =BLDLCD ?ST=1  BitsOK          Display already built correctly?
    108 0189F 1F           GOYES  PUTSTj          Yes, then just return
    109 018A1 873   BLDDS* ?ST=1  NoCur           Need to keep cursor in display?
    110 018A4 D2           GOYES  BLD004          No, then don't check if it is
    111 018A6 7FB6         GOSUB  DO=FC
    112 018AA 142          A=DATO A               Read FIRSTC & CURSOR
    113 018AD D8           B=A    A               B(B)=FIRSTC
    114 018AF F4           ASR    A
    115 018B1 F4           ASR    A               A(B)=CURSOR
    116 018B3 9E4          ?A<B   B               Is cursor before FIRSTC?
    117 018B6 81           GOYES  BLD003          Yes, then set FIRSTC=CURSOR
    118 018B8 1937         DO=(2) =WINDLN
    119 018BC 14E          C=DATO B               Read (WINDLN)
    120 018BF 19C7         DO=(2) =FIRSTC
    121 018C3 B6A          A=A-C  B               A(B)=(CURSOR)-(WINDLN)
    122 018C6 4A0          GOC    BLD004          (CURSOR)<(WINDLN) implies
    123                                             (CURSOR)<(FIRSTC)+(WINDLN)
    124 018C9 9EC          ?A<=B  B               Is cursor past (FIRSTC)+(WINDLN)?
    125 018CC 50           GOYES  BLD004          No, then skip
    126 018CE 148   BLD003 DATO=A B               Write out updated FIRSTC
    127 018D1 850   BLD004 ST=1   RArrow          Flag: Right arrow on
    128
    129             * The following code does not call SFLAG? since it is not
    130             * in the first ROM and is in the path to ROM text.
    131 018D4 7653         GOSUB  fCALC?          CALC flag set? (carry set="yes")
    132 018D8 7D86         GOSUB  DO=FC
    133 018DC D0           A=0    A
    134 018DE 14A          A=DATO B               A[A]=FIRSTC
    135 018E1 512          GONC   BLD005          Go if not CALC mode
    136 018E4 34B7         LC(5)  (=DSPBFS)-5     Left-arrow suppress for CALC
            4F2
    137 018EB C2           C=A+C  A
    138 018ED C2           C=A+C  A
    139 018EF 134          DO=C                   Point 5 nibs before (FIRSTC)
    140 018F2 142          A=DATO A               Read 2-1/2 bytes
    141 018F5 3420         LCHEX  20202
            202
    142 018FC 8A6          ?A#C   A               Blanks before first column?
    143 018FF 40           GOYES  BLD005          No.
    144 01901 D0           A=0    A               Yes, suppress left arrow
    145 01903 1B00  BLD005 DO=(5) =ANNAD1
            1E2
    146 0190A 14E          C=DATO B               Read ann column
    147 0190D 0B           CSTEX                  Swap ann bits into status reg
    148 0190F 843          ST=0   LeftA           Turn off left arrow bit
    149 01912 968          ?A=0   B               Does display start at char 0
    150 01915 50           GOYES  BLD006          Yes, then turn left arrow off
    151 01917 853          ST=1   LeftA           No, then turn left arrow on
    152 0191A 0B    BLD006 CSTEX                  Swap ann bits back into C(B)
    153 0191C 14C          DATO=C B               Write out annunciators
    154 0191F 7646         GOSUB  DO=FC
    155 01923 D0           A=0    A
    156 01925 14A          A=DATO B               A[A]=(FIRSTC)
    157 01928 31F5         LC(2)  95
```

```
158 0192C AE5          B=C    B
159 0192F 188          DO=DO- (FIRSTC)-(WINDLN)
160 01932 14E          C=DATO B              Read window length-1.  [0..21]
161 01935 B61          B=B-C  B              B=Position of last legal FIRSTC
162 01938 9E4          ?A<B   B              If A is less than this...
163 0193B 80           GOYES  BLD020         ...then okay
164 0193D AE4          A=B    B              Else use highest legal value
165 01940 840          ST=0   RArrow         Flag:  Right arrow off
166 01943 816   BLD020 CSRC
167 01946 816          CSRC                  C(14,15)=Window length-1
168 01949 181          DO=DO- 2              Point at WINDST
169 0194C D2           C=0    A
170 0194E 14E          C=DATO B              Read (WINDST)
171 01951 AF7          D=C    W              D(14,15)=Len cntdwn, D(A)=Windst
172 01954 3408         LC(5)  =DSPBFS
            4F2
173 0195B C2           C=C+A  A
174 0195D C2           C=C+A  A              C(A) points into display buffer
175 0195F 7950         GOSUB  BLDBIT         Build bit pattern in LCD
176 01963 1EE4         D1=(4) =ANNAD4
            3E
177 01969 14F          C=DAT1 B              Read ann column
178 0196C 0B           CSTEX                 Swap ann bits into status reg
179 0196E 844          ST=0   RightA         Turn off right arrow bit
180 01971 0B           CSTEX                 Swap ann bits back into C(B)
181 01973 860          ?ST=0  RArrow         Can right arrow be on?
182 01976 11           GOYES  BLD080         No, then skip
183 01978 14A          A=DATO B              Read next char in buffer
184 0197B 968          ?A=0   B              Is next char null?
185 0197E 90           GOYES  BLD080         Yes, then leave right arrow off
186 01980 0B           CSTEX                 Swap ann bits into status reg
187 01982 854          ST=1   RightA         Turn on right arrow bit
188 01985 0B           CSTEX                 Swap ann bits back into C(B)
189 01987 14D   BLD080 DAT1=C B              Write out annunciators
190 0198A 1DFF         D1=(2) =DD1CTL        Display control nibble
191 0198E 309          LCHEX  9
192 01991 1550         DAT1=C P              Turn on display w/timer on
193 01995 851          ST=1   BitsOK         Display is now built correctly
194 01998 842          ST=0   Phase          Cursor initially in ON phase
195                                          Also turns off TimOut
196 0199B 6A41         GOTO   DSPU01
197             *-
198             *-
```

```
                        STITLE BLDBIT
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) BLDBIT  -  Build Bit Patterns in Display
                **
                ** Category:   DSPUTL
                **
                ** Purpose:
                **      Used to put a given number of character's bit patterns
                **      in display given an arbitrary ASCII buffer.
                **
                ** Entry:
                **      P       =  0
                **      D(A)=Display starting position  (ie WINDST)
                **      D(14,15)=Number of positions to display minus 1
                **      C(A) points to buffer of characters
                **
                ** Exit:
                **      P       =  0
                **
                ** Calls:      IOFNDO
                **
                ** Uses.......
                **  Inclusive: A(W),B(W),C(W),D(W),D0,D1
                **
                ** Stk lvls:   1
                **
                ** Algorithm:
                **      For each character to be displayed
                **         If the high bit is on then
                **            Look for an alternate charset buffer.
                **            If one is found then
                **               Check for indirect character set and
                **                  change pointers if found
                **               Multiply character number by 12
                **               If this number is less than the length
                **                     of the charset buffer then use that
                **                     buffer
                **               else use the default bit pattern table
                **            else use the default bit pattern table
                **         else use the default pattern table
                **         If using the default bit pattern table
                **             then multiply the character number by 10
                **         Add the offset (char number times 10 or 12)
                **                to the start of the table being used
                **                and read in bit pattern.
                **         Read 3 nibble table entry from LCDTAB
                **         Double table entry to set carry if this
                **             char crosses a display driver boundary
                **             and to generate the lower 3 nibbles
                **             of the starting address of this display
                **             position.
                **         Write out bit pattern to display.
                **         If display driver boundary is crossed then
```

```
                **              shift the bit pattern 4 columns and
                **              move to next display driver and write
                **              out remainder of character.
                **
                ** History:
                **
                **    Date      Programmer          Modification
                **  --------    ----------  ---------------------------------
                **  10/19/82    B.S.        Added documentation
                **
                ********************************************************************
                ********************************************************************
0199F 301   =CHRBUF LC(1)  1
019A2 0E02          C=C&A  P
019A6 90A           ?C=0   P            Is the buffer length odd?
019A9 F0            GOYES  CHRB10       No, then use table
019AB 147           C=DAT1 A            Read indirect table address
019AE 135           D1=C
019B1 1533          A=DAT1 X            Read table length
019B5 172           D1=D1+ 3            Point past length
019B8 D9    CHRB10  C=B    A            C=2x
019BA 01            RTN
019BC 134   =BLDBIT DO=C
019BF D0    =BLDB10 A=0    A            A(A)=0
019C1 14A           A=DATO B            Read in character
019C4 161           DO=DO+ 2            Move pointer to next character
019C7 A64           A=A+A  B            (mod 128) then multiply by 2
019CA D8            B=A    A            B(A)=2x
019CC 531           GONC   BLDB20       Not alt chr.  Will fail next tst.
019CF 3200          LC(3)  =bALTCH
          0
019D4 8F00          GOSBVL =IOFNDO      Look for charset buffer
          000
019DB 440           GOC    BLDB20       Skip if buffer found
019DE D0            A=0    A            Will always fail next test
019E0 7BBF  BLDB20  GOSUB  CHRBUF
                                        C=2x
019E4 C5            B=B+B  A            B=4x
019E6 C5            B=B+B  A            B=8x
019E8 C1            B=B+C  A            B=10x
019EA C1            B=B+C  A            B=12x
019EC 2B            P=     11           12 nibbles per entry
019EE 8B0           ?B<A   A            Is this char within buffer?
019F1 D0            GOYES  BLDB30       Yes, handle it.
019F3 E1            B=B-C  A            No, B=10x
019F5 1F00          D1=(5) =ASCII       Point at normal bit pattern table
          000
019FC 29            P=     9            10 nibbles per entry
019FE 137   BLDB30  CD1EX               C(A)=Table pointer
01A01 C9            C=C+B  A            Add offset
01A03 136           CDOEX               C(A) saves DO, DO -> bit pattern
01A06 AD0   BLDB40  A=0    M            Clear upper nibs if normal char
01A09 1521          A=DATO WP           Read in bit pattern
01A0D 20            P=     0            Need to keep pointer at zero
01A0F 134           DO=C                Restore buffer pointer
```

```
    306 01A12 34C6          LC(5)  LCDTAB        Start of LCD address table
              A10
    307 01A19 CB            C=C+D  A
    308 01A1B CB            C=C+D  A
    309 01A1D CB            C=C+D  A^            C(A)=Address of LCD table entry
    310 01A1F E7            D=D+1  A             Increment position count
    311 01A21 135           D1=C                 Point at table entry
    312 01A24 3440          LC(5)  =DD3ST        Load upper 2 digits of LCD addr.
              1E2
    313 01A2B 1573          C=DAT1 X             Read table entry
    314 01A2F A36           C=C+C  X             Check for boundary.  Get addr.
    315 01A32 135           D1=C                 D1 is LCD address
    316 01A35 159B          DAT1=A 12            Write out 6 columns
    317 01A39 5C1           GONC   BLDB50        Skip if not driver boundry.
    318 01A3C 133           AD1EX
    319 01A3F AE0           A=0    B
    320 01A42 B24           A=A+1  XS
    321 01A45 131           D1=A                 Move D1 to next driver
    322 01A48 27            P=     7
    323 01A4A BF4   BLDB45  ASR    W             Loop to throw away 4 columns
    324 01A4D 0D            P=P-1
    325 01A4F 5AF           GONC   BLDB45
    326 01A52 1593          DAT1=A 4             Write out last 2 columns
    327 01A56 2E    BLDB50  P=     14
    328 01A58 A0F           D=D-1  P             Decrement length (lsd)
    329 01A5B 570           GONC   BLDB60        Borrow?  No, then skip
    330 01A5E 2F            P=     15
    331 01A60 A0F           D=D-1  P             Decrement next digit
    332 01A63 20    BLDB60  P=     0
    333 01A65 400           RTNC                 Rtn from BLDBIT if all chars done
    334 01A68 665F          GOTO   BLDB10        Loop back for next char
    335             *-
    336             *-
    337 01A6C 280   LCDTAB  CON(3) (4+0*12+#100)/2          Position 0
    338 01A6F 880           CON(3) (4+1*12+#100)/2                   1
    339 01A72 E80           CON(3) (4+2*12+#100)/2                   2
    340 01A75 490           CON(3) (4+3*12+#100)/2                   3
    341 01A78 A90           CON(3) (4+4*12+#100)/2                   4
    342 01A7B 0A0           CON(3) (4+5*12+#100)/2                   5
    343 01A7E 6A0           CON(3) (4+6*12+#100)/2                   6
    344 01A81 CA8           CON(3) (4+7*12+#100)/2+#800              7
    345 01A84 201           CON(3) (4+0*12+#200)/2                   8
    346 01A87 801           CON(3) (4+1*12+#200)/2                   9
    347 01A8A E01           CON(3) (4+2*12+#200)/2                   10
    348 01A8D 411           CON(3) (4+3*12+#200)/2                   11
    349 01A90 A11           CON(3) (4+4*12+#200)/2                   12
    350 01A93 021           CON(3) (4+5*12+#200)/2                   13
    351 01A96 621           CON(3) (4+6*12+#200)/2                   14
    352 01A99 C29           CON(3) (4+7*12+#200)/2+#800              15
    353 01A9C 281           CON(3) (4+0*12+#300)/2                   16
    354 01A9F 881           CON(3) (4+1*12+#300)/2                   17
    355 01AA2 E81           CON(3) (4+2*12+#300)/2                   18
    356 01AA5 491           CON(3) (4+3*12+#300)/2                   19
    357 01AA8 A91           CON(3) (4+4*12+#300)/2                   20
    358 01AAB 0A1           CON(3) (4+5*12+#300)/2                   21
```

```
359
360                *-
361                *-
362 01AAE 1BFF =NOTIME DO=(5) =DD1CTL        Point to timer control nibble
          3E2
363 01AB5 301          LCHEX  1
364 01AB8 15C0         DATO=C 1              Write out control nibble
365 01ABC 181          DO=DO- (DD1CTL)-(TIMER1)-5  Point at highest timer dgt
366 01ABF 307          LCHEX  7
367 01AC2 15C0  NOTIM1 DATO=C 1              Write out to timer
368 01AC6 15A0         A=DATO 1              Read back
369 01ACA 906          ?A#C   P              OK?
370 01ACD 5F           GOYES  NOTIM1         No.  Write again.
371 01ACF 854   NOTIM- ST=1   UpdOff         Inhibit cursor update
372 01AD2 65A0         GOTO   PUTSTA
373                *-
374                *-
375 01AD6 65CD  BLDLCj GOTO   BLDLCD
376                *-
377                *-
```

```
                STITLE DSPUPD
***************************************************************
***************************************************************
**
** Name:(S) DSPUPD  -  Display Update
**
** Category:   DSPUTL
**
** Purpose:
**      Process service request for display code.
**      Service request can be generated by TIMER1 and is used
**      either for:
**        1) Cursor blink, or
**        2) End of display delay.
**
** Entry:
**      P=0.
**
** Exit:
**      P=0.
**
** Calls:      GETSTA,D1=FC,BLDBIT,BLDB40,WRTTM1,SETSTA
**
** Uses.......
**  Inclusive: A(W),B(W),C(W),D(W),D0,D1,RAM(DSPSTA)
**
** Stk lvls:   2
**
** NOTE:
**      Saves contents of ST on entry into DSPSTA.  Restores
**      on exit.
**
** Algorithm:
**      Stores callers status bits in DSPSTA and recalls
**        display status.
**      Sets TimOut bit to indicate timer has timed out.
**      If UpdOff then display doesn't need updating so
**        set timer to a long time and return.
**      If CurOff then cursor is off and thus display
**        doesn't need updating so set timer to a long
**        time.  TimOut was set above which notes the
**        fact that the timer has timed out.  This is
**        used for display delay during line feed.
**      If BitsOK then the LCD reflects the display buffer
**        and doesn't require rebuilding just to change
**        cursor.  If not BitsOK then we need to rebuild
**        the LCD to make sure cursor will make sense, this
**        code will fall back through DSPUPD once display
**        has been updated.
**      Now need to change cursor.
**        The position of the cursor in the display is
**        calculated by looking at CURSOR, WINDST and WINDLN.
**        If the cursor isn't in display then set the timer
**          to a long time and return.
**        Depending on the Phase, either
```

```
                **          *  Rebuild the character that belongs in cursor
                **             position and toggle phase.
                **          *  Check if replace or insert cursor is required,
                **             build it in display, toggle phase and return.
                **
                ** History:
                **
                **    Date      Programmer          Modification
                **  --------    ----------  ---------------------------------
                **  02/25/83    NM          Added documentation
                **  06/07/83    B.S.        Enhanced documentation
                **
                ********************************************************************
                ********************************************************************
01ADA 75B0 =DSPUPD GOSUB  GETSTA
01ADE 85A  =DSPUP1 ST=1   TimOut          Indicate timed out
01AE1 874          ?ST=1  UpdOff
01AE4 AC           GOYES  NOTIME          Display timer has timed out but
                                          the cursor needn't be updated.
01AE6 876   DSPU01 ?ST=1  CurOff          Is cursor off?
01AE9 6E           GOYES  NOTIM-          Yes, then disable timer
01AEB 861   DSPU02 ?ST=0  BitsOK          Are display bits OK?
01AEE 8E           GOYES  BLDLCj          No, then build it
01AF0 7136         GOSUB  D1=FC
01AF4 14F          C=DAT1 B               C(B)=(FIRSTC)
01AF7 DO           A=0    A
01AF9 171          D1=D1+ 2
01AFC 14B          A=DAT1 B               A(B)=(CURSOR)
01AFF D8           B=A    A               Save cursor position
01B01 B6A          A=A-C  B               A(B)=(CURSOR)-(FIRSTC)
01B04 49A          GOC    NOTIME          Cursor not in display
01B07 1B37         DO=(5) =WINDLN
      4F2
01B0E 14E          C=DATO B               C(B)=(WINDLN)
01B11 9E6          ?A>C   B
01B14 A9           GOYES  NOTIME          Cursor not in display
01B16 181          DO=DO- (WINDLN)-(WINDST)
01B19 14E          C=DATO B               C(B)=(WINDST)
01B1C A62          C=C+A  B               C(B)=(WINDST)+(CURSOR)-(FIRSTC)
01B1F AF3          D=0    W
01B22 AE7          D=C    B               D(14,15)=0, D(A)=LCD position
01B25 862          ?ST=0  Phase           Is cursor phase on or off?
01B28 81           GOYES  DSPU40
01B2A 842          ST=0   Phase           Toggle phase on for next pass
01B2D 3408         LC(5)  =DSPBFS
      4F2
01B34 C9           C=C+B  A
01B36 C9           C=C+B  A               Calculate character address
01B38 708E         GOSUB  BLDBIT          Build char bit pattern
01B3C 6B10         GOTO   DSPU60          Jump
                *-
                *-
01B40 852   DSPU40 ST=1   Phase           Toggle phase off for next pass
01B43 1B00         DO=(5) =REPCUR         Assume replace cursor
      000
```

```
485 01B4A 867          ?ST=0  Insert         Is it really an insert cursor?
486 01B4D 50           GOYES  DSPU50         No, then skip
487 01B4F 169          DO=DO+ 10             Yes, move ptr to insert cursor
488 01B52 29   DSPU50  P=     9              10 nibbles per entry
489 01B54 7EAE         GOSUB  BLDB40         Build cursor
490 01B58 844  DSPU60  ST=0   UpdOff         Enable cursor update
491 01B5B D2           C=0    A
492 01B5D 3101         LCHEX  10
493 01B61 BF2  DSPTIM  CSL    W              Disp STATUS active on entry
494 01B64 8F00         GOSBVL =WRTTM1        Set Timer
          000
495 01B6B 166          DO=DO+ (DD1CTL)-(TIMER1)  Point to display control
496 01B6E 309          LCHEX  9              Disp & timer on
497 01B71 15C0         DATO=C 1              Write out display control
498 01B75 84A          ST=0   TimOut         Clear timeout bit
499 01B78 0B   PUTSTA  CSTEX                 Restore user status
500 01B7A 7F20         GOSUB  SETSTA
501 01B7E 0B           CSTEX
502 01B80 1B57 =USRSTA DO=(5) =DSPSTA
          4F2
503 01B87 0B           CSTEX
504 01B89 1563 PUTST1  C=DATO X              Recall original status bits
505 01B8D 0B           CSTEX                 Restore them
506 01B8F 20           P=     0
507 01B91 03           RTNCC
508             *-
509             *-
510 01B93 1B57 =GETSTA DO=(5) =DSPSTA
          4F2
511 01B9A 0B           CSTEX
512 01B9C 1543         DATO=C X
513 01BA0 1B87 =RCLSTA DO=(5) (=DSPSTA)+3
          4F2
514 01BA7 61EF         GOTO   PUTST1
515             *-
516             *-
517 01BAB 09   =STOSTA C=ST
518 01BAD 1B87  SETSTA DO=(5) (=DSPSTA)+3
          4F2
519 01BB4 1543         DATO=C X
520 01BB8 03           RTNCC
521             *-
522             *-
523             ************************************************************
524             ************************************************************
525             **
526             ** Name:(S) GETMSK  -  Get Mask for Character Protection Bitmap
527             **
528             ** Category:   DSPUTL
529             **
530             ** Purpose:
531             **      Point at location if protection bitmap and return a
532             **      mask for isolating bit corresponding to current cursor
533             **      position.
534             **
```

```
535                ** Entry:
536                **      P=0.
537                **
538                ** Exit:
539                **      P=0.
540                **      B[0]=C[0]=Mask nibble.
541                **      D0 points at nibble in bitmap for current cursor
542                **        position.  Mask can be used to isolate proper bit.
543                **
544                ** Calls:       None.
545                **
546                ** Uses.......
547                **              B[A], C, P, D0.
548                **
549                ** Stk lvls:    0
550                **
551                ** History:
552                **
553                **    Date      Programmer              Modification
554                **  --------    ----------   ----------------------------------
555                **  02/25/83    NM           Added documentation
556                **
557                ********************************************************************
558                ********************************************************************
559 01BBA 3475 =GETMSK LC(5)  (=DSPMSK)+23   Point to mask bits
           5F2
560 01BC1 D5           B=C    A              Copy mask pointer to B
561 01BC3 D2           C=0    A
562 01BC5 1BE7         D0=(5) =CURSOR
           4F2
563 01BCC 14E          C=DAT0 B              Read cursor position
564 01BCF C6           C=C+C  A
565 01BD1 C6           C=C+C  A              POS/4 to C[2-1], 4*rmd to C(0)
566 01BD3 80D0         P=C    0              Copy 4*remainder into P
567 01BD7 F6           CSR    A              Shift byte to C(B)
568 01BD9 FA           C=-C   A
569 01BDB C9           C=C+B  A              C=B-C  A   Calculate mask address
570 01BDD 134          D0=C                  D0=Mask address
571 01BE0 3C80         LCHEX  4000200010008  Magic constant for bit mask
           0010
           0020
           004
572 01BEF 20           P=     0              Clear pointer
573 01BF1 A85          B=C    P              Copy mask nibble to B(0)
574 01BF4 03           RTNCC
575                *-
```

```
                        STITLE BF2DSP  -  Buffer to Display
                *****************************************************************
                *****************************************************************
                **
                ** Name:    AVM2DS  -  Buffer to Display
                ** Name:(S) BF2DSP  -  Buffer to Display
                ** Name:    BF2DS+  -  Buffer to Display
                ** Name:    BF2DPP  -  Buffer to Display
                **
                ** Category:   DSPUTL
                **
                ** Purpose:
                **      AVM2DS:  Send buffer at AVMEMS to display
                **      BF2DPP:  Send PROMPT to display
                **      BF2DSP:  Send buffer at D1 to display
                **      BF2DS+:  Send buffer at (D1) to display
                **
                ** Entry:
                **      P      =  0
                **      BF2DPP:  Set D1 @ PROMPT
                **      BF2DSP:  D1 points at first char of buffer
                **      BF2DS+:  D1 points at address of start of buffer addr
                **      AVM2DS:  none
                **
                ** Exit:
                **      P      =  0
                **      Carry set
                **
                ** Calls:
                **      DSPCHA
                **
                ** Uses.......
                **  Exclusive: D1,C(A),A(B)
                **  Inclusive: A(W),B(W),C(W),D(W),D0,D1
                **
                ** Stk lvls:   4
                **
                ** Detail:
                **      In each case above the buffer is terminated by an FF
                **      byte.
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------   ----------------------------------
                **  10/19/82   B.S.         Updated documentation
                **
                *****************************************************************
                *****************************************************************
01BF6 1F00      =BF2DPP D1=(5) =PROMPT        Send Prompt to display buffer
      000
01BFD 6010              GOTO   BF2DSP
01C01 1F49      =AVM2DS D1=(5) =AVMEMS        Point D1 to input buffer pointer
      5F2
01C08 147       =BF2DS+ C=DAT1 A
```

```
629 01C0B 135          D1=C                Set D1 to start of buffer
630 01C0E       =BF2DSP
631 01C0E 14B          A=DAT1 B            Read char from buffer
632 01C11 AE6          C=A    B
633 01C14 A66          C=C+C  B            Is char >= 80
634 01C17 400          RTNC                Yes, then return
635 01C1A 137          CD1EX
636 01C1D 06           RSTK=C              Save D1
637 01C1F 7B10         GOSUB  DSPCHA       Send character to display
638 01C23 07           C=RSTK
639 01C25 135          D1=C                Restore D1
640 01C28 171          D1=D1+ 2            Move to next char
641 01C2B 52E          GONC   BF2DSP       (B.E.T.) Loop back for next char
642             *-
643             *-
644             ********************************************************************
645             ********************************************************************
646             **
647             ** Name:    fCALC?  -  Test if in CALC mode
648             **
649             ** Category:   GENUTL =  Generic Utilities (general purpose)
650             **
651             ** Purpose:
652             **      Test CALC mode system flag
653             **
654             ** Entry:
655             **      No necessary conditions
656             **
657             ** Exit:
658             **      D0=(5) (=SYSFLG)+11
659             **        Carry set = flCALC set
660             **        Carry clear = flCALC not set
661             **
662             ** Calls:      None
663             **
664             ** Uses.......
665             **  Exclusive: D0, A(A)
666             **
667             ** Stk lvls:   None
668             **
669             ** History:
670             **
671             **    Date     Programmer             Modification
672             **  --------   ----------   ----------------------------------
673             **  12/07/82   M.B.         Created routine
674             **
675             ********************************************************************
676             ********************************************************************
677 01C2E 1B4E =fCALC? D0=(5) (=SYSFLG)+11   Point at CALC flag nibble
          6F2
678 01C35 142          A=DAT0 A
679 01C38 C4           A=A+A  A            Set carry if CALC mode
680 01C3A 01           RTN
```

```
681                     STITLE DSPCHA/C  -  Display Character
682                 ********************************************************************
683                 ********************************************************************
684                 **
685                 ** Name:(S) DSPCHA  -  Display Character
686                 ** Name:(S) DSPCHC  -  Display Character
687                 **
688                 ** Category:   DSPUTL
689                 **
690                 ** Purpose:
691                 **      Accepts a byte for pseudo-device display driver.
692                 **      The routines take data from A or C and send the
693                 **      character to the display.
694                 **
695                 ** Entry:
696                 **      P       =  0
697                 **      DSPCHA:  A(B) contains character
698                 **      DSPCHC:  C(B) contains character
699                 **
700                 ** Exit:
701                 **      P       =  0
702                 **      Carry clear
703                 **
704                 ** Calls:       BLDDS*, BLDDSP, CKSREQ, CLEARD, D0=CUR, D0=FC,
705                 **              DSPCH., DSPTIM, FINDA, GETMSK, GETSTA, MOVC00,
706                 **              MOVCUR, MOVED3, MOVEU3, NOKEYS, PUTSTA, R<RSTK,
707                 **              RCLSTA, RSTK<R, SCNRT, SCRLLR, SETFCA, SLEEP,
708                 **              cksreq, TBLJMC, USRSTA.
709                 **
710                 ** Uses.......
711                 **  Inclusive: A(W),B(W),C(W),D(W),D0,D1,RAM(See note below)
712                 **
713                 ** Stk lvls:   2
714                 **
715                 ** NOTE:
716                 **      This routine will call CKSREQ if CR or LF is sent.
717                 **      This implies that a poll may happen.  That will cause
718                 **      certain RAM locations (SNAPSHOT) to be altered.
719                 **      This routine uses R<RSTK / RSTK<R to preserve stack
720                 **      levels--this also uses RAM.
721                 **
722                 **      This routine may also transfer control out to HPIL.
723                 **      The HPIL ROM may not have exactly the same register
724                 **      usage for a given character, ie don't assume a certain
725                 **      character will leave a certain register preserved
726                 **      just because this code doesn't seem to use it.  For
727                 **      any character, A,B,C,D,D0,D1 may be used.
728                 **
729                 ** Detail:
730                 **      This routine provides the mechanism to access the
731                 **      pseudo-device that controls the display.
732                 **      This device has 3 nibbles of status that are
733                 **      defined as follows:
734                 **        S0 -- Miscellaneus uses
735                 **        S1 -- Set iff LCD currently matches display buffer
```

```
736                **        S2 -- Cursor blink phase
737                **        S3 -- Display needn't contain cursor
738                **        S4 -- Disable cursor update
739                **        S5 -- Display buffer needs to be cleared(1)
740                **        S6 -- Cursor on(0)/off(1)
741                **        S7 -- Insert(1)/Replace(0) mode
742                **        S8 -- Cursor/FirstC need to be cleared(1)
743                **        S9 -- Suppress Delay(1)   (Auto-clears)
744                **        S10 -- Display has timed out
745                **
746                **      The pseudo-device accepts the following escape
747                **      sequences:
748                **        Esc Q -- Insert cursor
749                **        Esc R -- Replace cursor
750                **        Esc C -- Cursor right
751                **        Esc D -- Cursor left
752                **        Esc H -- Home cursor
753                **        Esc J -- Clear Display
754                **        Esc K -- Delete through end of line
755                **        Esc > -- Cursor on
756                **        Esc < -- Cursor off
757                **        Esc E -- Reset display
758                **        Esc P -- Delete char
759                **        Esc % <col> <row> -- Set cursor position absolute
760                **        Esc Ctrl-C -- Cursor far right
761                **        Esc Ctrl-D -- Cursor far left
762                **
763                **
764                ** History:
765                **
766                **    Date     Programmer             Modification
767                **  --------   ----------   ---------------------------------
768                **  10/19/82   B.S.         Updated documentation
769                **  02/25/83   NM           Updated "CALLS" section
770                **
771                ****************************************************************
772                ****************************************************************
773 01C3C DA    =DSPCHC A=C     A               Copy C(B) to A(B)
774 01C3E 25    =DSPCHA P=      SavLvl          Number of stack levels to save
775 01C40 7000          GOSUB   =R<RSTK         Save n stack levels
776 01C44 7800          GOSUB   DSPCH.
777 01C48 25            P=      SavLvl          Number of stack levels to restore
778 01C4A 7000          GOSUB   =RSTK<R         Restore n stack levels
779 01C4E 03            RTNCC                   GOSBVL/RTNCC can't be packed out!
780                *-
781                *-
782 01C50 7F3F  DSPCH.  GOSUB   GETSTA
783 01C54 3427          LC(5)   DSPCH*
          C10
784 01C5B 06            RSTK=C
785 01C5D 1B47          DO=(5)  =DSPCHX
          6F2
786 01C64 146           C=DATO  A
787 01C67 8AA           ?C=0    A
788 01C6A 60            GOYES   DSPCH@
```

```
789 01C6C 06           RSTK=C
790 01C6E 03           RTNCC
791             *-
792             *-
793 01C70 07    DSPCH@ C=RSTK
794 01C72 843   DSPCH* ST=0   NoCur          Keep cursor in display
795 01C75 20           P=     0
796 01C77 1BB7         DO=(5) =ESCSTA        Point to ESCSTA
          4F2
797 01C7E 1560         C=DATO P
798 01C82 70A7         GOSUB  TBLJMC
799 01C86 6C1          REL(3) DSPS-0
800 01C89 A72          REL(3) DSPS-1
801 01C8C 900          REL(3) DSPS-2
802 01C8F AC2          REL(3) DSPS-3
803             *-
804             *-
805             *-
806             *-
807 01C92 B6A   DSPS20 A=A-C  B
808 01C95 3106  DSPS-2 LC(2)  96
809 01C99 9EE          ?A>=C  B              Is column > 96?
810 01C9C 6F           GOYES  DSPS20         Yes, then reduce by 96
811 01C9E 162          DO=DO+ (CURSOR)-(ESCSTA)
812 01CA1 148          DATO=A B
813 01CA4 D8           B=A    A              Hold (CURSOR) in B
814 01CA6 79E3         GOSUB  DO=CUR
815 01CAA 3102         LCASC  \ \
816 01CAE 14A   DSPS21 A=DATO B
817 01CB1 96C          ?A#0   B
818 01CB4 E0           GOYES  DSPS22
819 01CB6 14C          DATO=C B
820 01CB9 181          DO=DO- 2
821 01CBC A6D          B=B-1  B
822 01CBF 5EE          GONC   DSPS21
823 01CC2 841   DSPS22 ST=0   BitsOK
824 01CC5 303          LC(1)  EscSt3
825 01CC8 6471         GOTO   ESCSET
826             *-
827             *-
828 01CCC 72F3  DSP-CR GOSUB  CLEARD         Clear display if needed
829 01CD0 858          ST=1   ResetC         Set Reset Cursor flag
830                                           so that when line feed sets
831                                           Clear flag it will also
832                                           reset CURSOR and FIRSTC
833 01CD3 853          ST=1   NoCur          For scrolling
834 01CD6 876          ?ST=1  CurOff         Is cursor off?
835 01CD9 A0           GOYES  DSPCR1         Yes, then build display
836 01CDB 60CB         GOTO   BLDLCD         Build display and return
837             *-
838             *-
839 01CDF 70BE =DOSCRL GOSUB  GETSTA
840 01CE3 879   DSPCR1 ?ST=1  XDelay         Is delay suppress on?
841 01CE6 44           GOYES  DSPC33         Yes.  Act like delay = inf.
842 01CE8 866          ?ST=0  CurOff         Is cursor on?
```

```
    843 01CEB F3          GOYES  DSPC33         Yes.  Build disp w/cursor in it
    844 01CED 853         ST=1   NoCur          Set NoCur in case of DOSCRL entry
    845 01CF0 1A64        DO=(4) =SCROLT
              9F
    846 01CF6 14A         A=DATO B              Read scroll delay time
    847 01CF9 96C         ?A#0   B              Is it zero?
    848 01CFC 82          GOYES  DSPCR3         No, then check for other values
    849 01CFE 1A37        DO=(4) =WINDLN
              4F
    850 01D04 14E         C=DATO B              Read window len-1
    851 01D07 7F93        GOSUB  A=CUR          Read cursor position
    852 01D0B B6A         A=A-C  B              Calculate first char in window
    853 01D0E A6C         A=A-1  B              Use position one before cursor
    854                                         since cursor is off
    855 01D11 3106        LC(2)  96
    856 01D15 9E2         ?A<C   B              Would this be before char 0?
    857 01D18 50          GOYES  DSPCR2         No, then okay
    858 01D1A AE0         A=0    B              Yes, then just start at char 0
    859 01D1D 181   DSPCR2 DO=DO- (CURSOR)-(FIRSTC)
    860 01D20 6DAB        GOTO   BLD003         Yes, Update FIRSTC,Build disp,Rtn
    861             *-
    862             *-
    863 01D24 B64   DSPCR3 A=A+1  B              Is it an infinity(FF)?
    864 01D27 560         GONC   DSPC35         No, then set timer
    865 01D2A 617B  DSPC33 GOTO   BLDLCD        Yes, then build display,rtn
    866             *-
    867             *-
    868 01D2E 7B37  DSPC35 GOSUB  NOKEY+
    869 01D32 7B6B  DSPCR4 GOSUB  BLDDS*
    870             * Build display, restore user's status.  This routine will
    871             * clear Phase and set UpdOff but will leave the timer running.
    872             * When the display timer expires,  CKSREQ will call DSPUPD
    873             * which will see that UpdOff is set and then set Phase and set
    874             * the display timer not to go off again soon.
    875 01D36 766E        GOSUB  RCLSTA         Recall display status
    876 01D3A 1B37        DO=(5) =WINDLN
              4F2
    877 01D41 14E         C=DATO B
    878 01D44 168         DO=DO+ (FIRSTC)-(WINDLN)
    879 01D47 D0          A=0    A
    880 01D49 14A         A=DATO B
    881 01D4C E4          A=A+1  A
    882 01D4E D8          B=A    A
    883 01D50 A61         B=B+C  B
    884 01D53 31F5        LC(2)  95
    885 01D57 9E1         ?B>C   B              At end of display?
    886 01D5A B4          GOYES  DSPCR7         Yes, then exit
    887 01D5C 3408        LC(5)  =DSPBFS
              4F2
    888 01D63 C9          C=C+B  A
    889 01D65 C9          C=C+B  A
    890 01D67 134         DO=C                  DO points to last char in window
    891 01D6A 14E         C=DATO B              Read buffer char
    892 01D6D 96A         ?C=0   B              Is it a null?
    893 01D70 53          GOYES  DSPCR7         Yes, then exit
```

```
894 01D72 7036          GOSUB  SETFCA          Update FIRSTC
895 01D76 841           ST=0   BitsOK
896 01D79 D2            C=0    A
897 01D7B 1A64          DO=(4) =SCROLT
          9F
898 01D81 14E           C=DATO B               Read (SCROLT)
899 01D84 79DD          GOSUB  DSPTIM          Start scroll timer & rcl user sta
900 01D88       DSPCR5                         User's status must be active here
901 01D88 7D10          GOSUB  cksreq          Check service requests
902 01D8C 701E          GOSUB  RCLSTA          Restore status
903 01D90 87A           ?ST=1  TimOut          Has timer expired?
904 01D93 F9            GOYES  DSPCR4          Yes, then scroll
905 01D95 77ED          GOSUB  USRSTA          Recall user's status
906 01D99 8E00          GOSUBL =SLEEP          No, then goto sleep
          00
907 01D9F 500           RTNNC                  Return with cry clr if buffer not
908                                             empty.  User's status are active
909 01DA2 45E           GOC    DSPCR5          (B.E.T.) Bfr empty; cont wait
910             *-
911             *-
912 01DA5 6ADD  DSPCR7 GOTO   USRSTA
913             *-
914             *-
915 01DA9 8C00  cksreq GOLONG =CKSREQ
          00
916             *-
917             *-
918             * DSPDLY causes a display delay (if cursor off)
919             *     Stack levels:  5
920             *
921 01DAF 70ED =DSPDLY GOSUB  GETSTA
922 01DB3 6E10          GOTO   DSPLFO
923             *-
924             *-
925 01DB7 633D  DSPU2j GOTO   DSPU02
926             *-
927             *-
928 01DBB 71CD  DSP-LF GOSUB  USRSTA          Recall user status
929 01DBF 76EF          GOSUB  cksreq          Check service requres
930 01DC3 79DD          GOSUB  RCLSTA          Restore status
931 01DC7 855           ST=1   Clear           Need display cleared later
932 01DCA 853           ST=1   NoCur           For scrolling
933 01DCD 879           ?ST=1  XDelay          Is delay suppressed?
934 01DD0 36            GOYES  NOTIMX          Yes, then don't delay
935 01DD2 866   DSPLFO ?ST=0  CurOff          Is cursor on?
936 01DD5 2E            GOYES  DSPU2j          Yes, then don't delay
937 01DD7 D2            C=0    A
938 01DD9 1B84          DO=(5) =DELAYT
          9F2
939 01DE0 14E           C=DATO B               Read delay time
940 01DE3 96A           ?C=0   B               Is delay zero?
941 01DE6 05            GOYES  NOTIMj          Yes, then don't delay
942 01DE8 1B24          DO=(5) =ATNFLG
          4F2
943 01DEF 1520          A=DATO P
```

```
944 01DF3 90C          ?A#0   P            ATNFLG set?
945 01DF6 04           GOYES  NOTIMj       Yes.  No delay.
946 01DF8 7176         GOSUB  NOKEY+       Flush keybuffer before delay
947 01DFC DA           A=C    A            Copy delay time
948 01DFE B64          A=A+1  B            Is it infinity(FF)?
949 01E01 5C0          GONC   DSPLF2       No, no scroll.
950 01E04 707D         GOSUB  PUTSTA       Restore user status
951 01E08 7223 DSPLF1  GOSUB  SCRLLR       Infinite scroll
952 01E0C 03           RTNCC               Return from DSPCHA
953             *-
954             *-
955 01E0E 7F4D DSPLF2  GOSUB  DSPTIM       Set display timer & rcl user sta
956 01E12 728A         GOSUB  BLDDSP       Build display
957 01E16 7F8F DSPLF3  GOSUB  cksreq       Check service request
958 01E1A 728D         GOSUB  RCLSTA       Restore status
959 01E1E 87A          ?ST=1  TimOut       Timed out?
960 01E21 48           GOYES  DSPCR7       Yes.  Restore user status & exit
961 01E23 795D         GOSUB  USRSTA       Restore user status
962 01E27 8E00         GOSUBL =SLEEP       Sleep
          00
963 01E2D 500          RTNNC               Return if keys in buffer
964 01E30 45E          GOC    DSPLF3       What happened?
965             *-
966             *-
967 01E33 849  NOTIMX   ST=0   XDelay       Clear delay suppression
968 01E36 677C NOTIMj   GOTO   NOTIME
969             *-
970             *-
971 01E3A 301  DSP-EC   LC(1)  EscSt1       Set status:  ESC was received.
972 01E3D 1BB7 ESCSET  DO=(5) =ESCSTA
          4F2
973 01E44 15C0         DATO=C 1
974 01E48 6F2D PUTSj2   GOTO   PUTSTA
975             *-
976             *-
977             *-
978 01E4C 7395 DSPS-O   GOSUB  FINDA
979 01E50 D0           CON(2) 13           Carriage Return
980 01E52 A7E          REL(3) DSP-CR
981 01E55 A0           CON(2) 10           Line Feed
982 01E57 46F          REL(3) DSP-LF
983 01E5A B1           CON(2) 27           Escape
984 01E5C EDF          REL(3) DSP-EC
985 01E5F 80           CON(2) 8            Back Space
986 01E61 061          REL(3) DSP-BS
987 01E64 00           CON(2) 00           Otherwise fall thru to ADDCHR
988 01E66 7852 ADDCHR  GOSUB  CLEARD       Clear display if needed
989 01E6A 968          ?A=0   B            Is it a null?
990 01E6D BD           GOYES  PUTSj2       Yes, then ignore it
991 01E6F 1AA4         DO=(4) =NEEDSC
          9F
992 01E75 30F          LCHEX  F
993 01E78 15C0         DATO=C 1            Set NEEDSCroll flag
994 01E7C 841          ST=0   BitsOK       LCD not updated
995 01E7F 867          ?ST=0  Insert       Insert mode?
```

```
 996 01E82 07          GOYES  ADDC20        No, then find cursor position
 997 01E84 7134        GOSUB  SCNRT         Yes, then make room for character
 998 01E88 456         GOC    ADDC17        Buffer full?  Yes, then exit.
 999 01E8B 137         CD1EX                C=D1
1000 01E8E DF          CDEX   A             C(A)=Start of dest,D(A)=Old D1
1001 01E90 131         D1=A                 D1=End of dest
1002 01E93 EE          C=A-C  A             Length to move
1003 01E95 14B         A=DAT1 B             Read a byte
1004 01E98 96C         ?A#0   B             Is it a null?
1005 01E9B 41          GOYES  ADDC05        No, then D1 is end of dest
1006 01E9D 1C1         D1=D1- 2             Yes, then look at byte before it
1007 01EA0 14B         A=DAT1 B
1008 01EA3 968         ?A=0   B             Is it a null?
1009 01EA6 F4          GOYES  ADDC25        Yes, then just replace char
1010 01EA8 173         D1=D1+ 4             Increment end of dest
1011 01EAB E6          C=C+1  A
1012 01EAD E6          C=C+1  A             Increment length to move
1013 01EAF 137   ADDC05 CD1EX
1014 01EB2 134         DO=C
1015 01EB5 137         CD1EX                D1=End Dest,C(A)=Length to move
1016 01EB8 181         DO=DO- 2             DO=End of Source
1017 01EBB 8E00        GOSUBL =Moved3       Move text following insert
          00
1018 01EC1 DB          C=D    A
1019 01EC3 135         D1=C                 Restore original D1
1020 01EC6 AE4   ADDC10 A=B    B            Recall char to be inserted
1021 01EC9 148         DAT0=A B             Write out new char
1022 01ECC 866         ?ST=0  CurOff
1023 01ECF 21          GOYES  ADDC15
1024 01ED1 75EC        GOSUB  GETMSK
1025 01ED5 1520        A=DAT0 P             Read mask nibble
1026 01ED9 0E0E        A=A!C  P             OR in bit
1027 01EDD 1500        DAT0=A P             Update mask nibble
1028 01EE1 840   ADDC15 ST=0   CurLft
1029 01EE4 85B         ST=1   NoChFC        Don't change FIRSTC
1030 01EE7 7654        GOSUB  MOVC00        Move cursor right
1031 01EEB 843         ST=0   NoCur
1032 01EEE 698C  ADDC17 GOTO   PUTSTA
1033             *-
1034             *-
1035 01EF2 AE8   ADDC20 B=A    B
1036 01EF5 7A91  ADDC25 GOSUB  DO=CUR       Point DO at cursor position
1037 01EF9 5CC         GONC   ADDC10        (B.E.T.)
1038             *-
1039             *-
1040 01EFC 857   D-ESCQ ST=1   Insert        Turn on insert cursor
1041 01EFF 6950        GOTO   ESCSTO
1042             *-
1043             *-
1044
1045 01F03 7CD4  DSPS-1 GOSUB  FINDA
1046 01F07 15          CON(2) \Q\
1047 01F09 3FF         REL(3) D-ESCQ        Esc Q -- Insert cursor
1048 01F0C E4          CON(2) \N\
1049 01F0E EEF         REL(3) D-ESCQ        Esc N -- Insert cursor(with wrap)
```

```
1050 01F11 25          CON(2) \R\
1051 01F13 050         REL(3) D-ESCR        Esc R -- Replace cursor
1052 01F16 34          CON(2) \C\
1053 01F18 A50         REL(3) D-ESCC        Esc C -- Cursor right
1054 01F1B 44          CON(2) \D\
1055 01F1D 4A0         REL(3) D-ESCD        Esc D -- Cursor left
1056 01F20 84          CON(2) \H\
1057 01F22 9A0         REL(3) D-ESCH        Esc H -- Home cursor
1058 01F25 A4          CON(2) \J\
1059 01F27 5D0         REL(3) D-ESCJ        Esc J -- Clear display
1060 01F2A B4          CON(2) \K\
1061 01F2C 0D0         REL(3) D-ESCK        Esc K -- Delete thru end of line
1062 01F2F E3          CON(2) \>\
1063 01F31 BA0         REL(3) D-ESC>        Esc > -- Cursor on
1064 01F34 C3          CON(2) \<\
1065 01F36 DB0         REL(3) D-ESC<        Esc < -- Cursor off
1066 01F39 54          CON(2) \E\
1067 01F3B 1F0         REL(3) D-ESCE        Esc E -- Reset display
1068 01F3E 05          CON(2) \P\
1069 01F40 001         REL(3) D-ESCP        Esc P -- Delete char
1070 01F43 F4          CON(2) \O\
1071 01F45 BF0         REL(3) D-ESCP        Esc O -- Delete char (with wrap)
1072 01F48 52          CON(2) \%\
1073 01F4A E31         REL(3) D-ESC%        Esc % -- Set cursor absolute
1074 01F4D 30          CON(2) 3
1075 01F4F E30         REL(3) D-ES.C        Esc <Ctrl-C> -- Cursor far right
1076 01F52 40          CON(2) 4
1077 01F54 650         REL(3) D-ES.D        Esc <Ctrl-D> -- Cursor far left
1078
1079 01F57 00          NIBHEX 00
1080 01F59       ESCST0
1081 01F59       DSPS-3                      {ESC % <Col> <Row> should ignore
1082 01F59 300         LC(1)  EscSt0        row}
1083 01F5C 841         ST=0   BitsOK
1084 01F5F 6DDE        GOTO   ESCSET
1085             *-
1086             *-
1087 01F63 847   D-ESCR ST=0   Insert        Turn on replace cursor
1088 01F66 52F         GONC   ESCST0        (B.E.T.)
1089             *-
1090             *-
1091 01F69 1BC7  DO=FC  DO=(5) =FIRSTC
           4F2
1092 01F70 01          RTN
1093             *-
1094             *-
1095 01F72 7C41  D-ESCC GOSUB  CLEARD        Clear display if needed
1096 01F76 7911        GOSUB  DO=CUR
1097 01F7A 14E         C=DATO B
1098 01F7D 96A         ?C=0   B
1099 01F80 9D          GOYES  ESCST0
1100 01F82 840         ST=0   CurLft
1101 01F85 75B3  ESCD10 GOSUB  MOVCUR
1102 01F89 6FCF  ESCSOj GOTO   ESCST0
1103             *-
```

```
1104                *-
1105 01F8D 7131   D-ES.C GOSUB  CLEARD
1106 01F91 7EF0   D-E.C1 GOSUB  DO=CUR
1107 01F95 14E           C=DATO B
1108 01F98 96A           ?C=0   B
1109 01F9B EE            GOYES  ESCS0j
1110 01F9D 840           ST=0   CurLft
1111 01FA0 7A93          GOSUB  MOVCUR
1112 01FA4 5CE           GONC   D-E.C1
1113 01FA7 41E           GOC    ESCS0j          (B.E.T.)
1114                *-
1115                *-
1116 01FAA 7411   D-ES.D GOSUB  CLEARD
1117 01FAE 850           ST=1   CurLft
1118 01FB1 D0            A=0    A
1119 01FB3 7FE3          GOSUB  SETFCA           Zero FIRSTC to speed up loop
1120 01FB7 7383   D-E.D1 GOSUB  MOVCUR
1121 01FBB 5BF           GONC   D-E.D1
1122 01FBE 4AC           GOC    ESCS0j          (B.E.T.)
1123                *-
1124                *-
1125 01FC1         DSP-BS
1126 01FC1 7DF0   D-ESCD GOSUB  CLEARD           Clear display if needed
1127 01FC5 850           ST=1   CurLft
1128 01FC8 4CB           GOC    ESCD10          (B.E.T.)
1129                *-
1130                *-
1131 01FCB 73F0   D-ESCH GOSUB  CLEARD           Clear display if needed
1132 01FCF D0            A=0    A
1133 01FD1 749F          GOSUB  DO=FC
1134 01FD5 1583          DATO=A 4                Clear FIRSTC and CURSOR
1135 01FD9 426           GOC    ESCSTj          (B.E.T.)
1136                *-
1137                *-
1138 01FDC 846    D-ESC> ST=0   CurOff           Turn cursor on
1139 01FDF 843           ST=0   NoCur            Keep cursor in display
1140 01FE2 1BB7  ESC>10  DO=(5) =ESCSTA
             4F2
1141 01FE9 D2            C=0    A
1142 01FEB 15C0          DATO=C 1                Return to escape state zero
1143 01FEF 61B8          GOTO   BLDDS*           build display
1144                *-
1145                *-
1146 01FF3 856    D-ESC< ST=1   CurOff           Turn cursor off
1147 01FF6 853           ST=1   NoCur            Don't keep cursor in display
1148 01FF9 58E           GONC   ESC>10           (B.E.T.)
1149                *-
1150                *-
1151 01FFC         D-ESCJ
1152 01FFC 72C0   D-ESCK GOSUB  CLEARD           Clear display if needed
1153 02000 75B2          GOSUB  SCNRT
1154 02004 D2            C=0    A                C(B)=Null
1155 02006 130           DO=A                    DO points past unprotected field
1156 02009 4E0           GOC    ESCK10           At display end?  Yes; fill nulls.
1157 0200C 14A           A=DATO B                No, then read byte
```

```
1158 0200F 968          ?A=0   B          Is this a null byte?
1159 02012 60           GOYES  ESCK10      Yes, then fill nulls
1160 02014 3102         LCASC  \ \         No, then fill blanks
1161 02018 181   ESCK10 D0=D0- 2           Move pointer back
1162 0201B 14C          DAT0=C B           Write out null or blank
1163 0201E 136          CDOEX
1164 02021 8B3          ?C<D   A           Done yet?
1165 02024 81           GOYES  ESCSTj      Yes, then exit
1166 02026 136          CDOEX              No, then do another byte
1167 02029 5EE          GONC   ESCK10      (B.E.T.)  Loop back
1168              *-
1169              *-
1170 0202C 847   D-ESCE ST=0   Insert
1171 0202F 846          ST=0   CurOff
1172 02032 855          ST=1   Clear
1173 02035 858          ST=1   ResetC
1174 02038 7680         GOSUB  CLEARD
1175 0203C 6C1F  ESCSTj GOTO   ESCSTO
1176              *-
1177              *-
1178 02040 7E70  D-ESCP GOSUB  CLEARD      Clear display if needed
1179 02044 840          ST=0   AtEnd
1180 02047 7E62         GOSUB  SCNRT       Look right for end of buf to be
1181                                        moved
1182 0204B 550          GONC   ESCP10      At end of buffer?
1183 0204E 850          ST=1   AtEnd       Yes, then indicated it with flag
1184 02051 137   ESCP10 CD1EX
1185 02054 DF           CDEX   A           D(A)=Old D1,C(A)=Start of Source
1186 02056 134          D0=C               D0=Start of source
1187 02059 135          D1=C
1188 0205C 1C1          D1=D1- 2           D1=Start of destination
1189 0205F EE           C=A-C  A           C(A)=Length to be moved
1190 02061 8E00         GOSUBL =MOveu3     Move memory
           00
1191 02067 AE2          C=0    B
1192 0206A 870          ?ST=1  AtEnd       Past end of memory?
1193 0206D E0           GOYES  ESCP20      Yes, then fill in null
1194 0206F 14E          C=DAT0 B           No, then read last character
1195 02072 96A          ?C=0   B           Is it a null?
1196 02075 60           GOYES  ESCP20      Yes, fill in a null before it
1197 02077 3102         LCASC  \ \         No, then fill in a blank.
1198 0207B 14D   ESCP20 DAT1=C B           Write out fill character
1199 0207E AEB          C=D    B
1200 02081 135          D1=C               Restore original D1
1201 02084 64DE         GOTO   ESCSTO
1202              *-
1203              *-
1204 02088 7630  D-ESC% GOSUB  CLEARD
1205 0208C 302          LC(1)  EscSt2
1206 0208F 6DAD         GOTO   ESCSET
1207              *-
1208              *-
1209 02093 7900 =D0=CUR GOSUB  CA=CUR
1210 02097 C2    D0=CAA C=C+A  A
1211 02099 C2          C=C+A  A
```

```
1212 0209B 134          DO=C
1213 0209E 03           RTNCC
1214            *-
1215            *-
1216 020A0 AF0   CA=CUR A=0     W           Clear A(5-0)
1217 020A3 3408         LC(5)   =DSPBFS
           4F2
1218 020AA 1BE7 =A=CUR  DO=(5)  =CURSOR
           4F2
1219 020B1 14A          A=DATO B            Read in cursor position
1220 020B4 03           RTNCC
1221            *-
1222            *-
1223            ****************************************************************
1224            ****************************************************************
1225            **
1226            ** Name:(S) DSPCL?  -  Clear display buffer if necessary
1227            **
1228            ** Category:   DSPUTL
1229            **
1230            ** Purpose:
1231            **      Clear display buffer if Clear bit set in display status
1232            **      Reset cursor position if ResetC bit set in display
1233            **      status
1234            **
1235            ** Entry:
1236            **      P =    0
1237            **
1238            ** Exit:
1239            **
1240            ** Calls:      GETSTA,CLEARD,PUTSTA
1241            **
1242            ** Uses.......
1243            **  Inclusive: A(M),C(B)
1244            **
1245            ** Stk lvls:   2
1246            **
1247            ** History:
1248            **
1249            **    Date      Programmer              Modification
1250            **  --------    ----------   ----------------------------------
1251            **  11/01/83    B.S.         Added documentation
1252            **
1253            ****************************************************************
1254            ****************************************************************
1255 020B6 79DA =DSPCL? GOSUB  GETSTA
1256 020BA 7400         GOSUB  CLEARD
1257 020BE 69BA         GOTO   PUTSTA
1258            *-
1259            *-
1260            *   This routine returns with carry set.  Uses A(M),C(B)
1261 020C2 865   CLEARD ?ST=0  Clear           Need to clear display buffer?
1262 020C5 34           GOYES  CLEAR2          No, then return
1263 020C7 845          ST=0   Clear           Clear flag
1264 020CA 3111         LC(2)  (96*2+24)/12-1 Loop counter
```

```
1265 020CE 1B08          DO=(5) =DSPBFS
           4F2
1266 020D5 841   CLEARO  ST=0   BitsOK
1267 020D8 AD0           A=0    M
1268 020DB 1505  CLEAR1  DATO=A M              Clear 12 nibbles
1269 020DF 16B           DO=DO+ 12             Move past these nibbles
1270 020E2 A6E           C=C-1  B              Decrement counter
1271 020E5 55F           GONC   CLEAR1         Loop back until done
1272 020E8 878           ?ST=1  ResetC
1273 020EB 22            GOYES  CLEAR3
1274 020ED D8            B=A    A
1275 020EF 77BF          GOSUB  A=CUR
1276 020F3 DC            ABEX   A              Restore A(B),B(B)=(CURSOR)
1277 020F5 3102          LCASC  \ \
1278 020F9 161   CLEAR.  DO=DO+ 2              Move DO from CURSOR to DSPBFS
1279                                           or move it to point to next char
1280 020FC A6D           B=B-1  B              Decrement counter
1281 020FF 400           RTNC                  Return with carry set when done
1282 02102 14C           DATO=C B              Change null to a space.
1283 02105 53F           GONC   CLEAR.         (B.E.T.) Loop back
1284             *-
1285             *-
1286 02108 868   CLEAR2  ?ST=0  ResetC         Need to reset cursor?
1287 0210B 00            RTNYES                No, then return
1288 0210D 848   CLEAR3  ST=0   ResetC         Reset Clear Cursor flag
1289 02110 755E          GOSUB  DO=FC
1290 02114 D2            C=0    A
1291 02116 15C3          DATO=C 4              Clear FIRSTC and CURSOR
1292 0211A 1A04          DO=(4) =DSPMSK
           5F
1293 02120 E6            C=C+1  A              Clear 24 nibbles (96 bits)
1294 02122 52B           GONC   CLEARO         (B.E.T.) Clear DSPMSK bits
1295             *-
1296             *-
1297 02125 1FC7  D1=FC   D1=(5) =FIRSTC
           4F2
1298 0212C 01            RTN
1299             *-
1300             *-
```

```
1301                    STITLE SCRLLR  -  Scroll Left and Right
1302                ****************************************************************
1303                ****************************************************************
1304                **
1305                ** Name:(S) SCRLLR  -  Scroll Left and Right
1306                **
1307                ** Category:   KEYUTL
1308                **
1309                ** Purpose:
1310                **      Watch for scroll keys and perform display scroll
1311                **
1312                ** Entry:
1313                **      P       =  0
1314                **
1315                ** Exit:
1316                **      P       =  0
1317                **      A(B) contains keycode that is first in key buffer
1318                **
1319                ** Calls:      ALMSRV, BLDDSP, BLDLCD, CKSREQ, D1=FC, FINDDO,
1320                **             GETSTA, POPBUF, RPTKY, SCRL60, SETFC, SETTM0,
1321                **             SLEEP, USRSTA.
1322                **
1323                ** Uses.......
1324                **  Exclusive:
1325                **  Inclusive: A(W),B(W),C(W),D(W),D0,D1
1326                **
1327                ** Stk lvls:   5
1328                **
1329                ** Detail:
1330                **      Sleeps and watches for scrolling key in the key
1331                **      buffer and causes the display to respond
1332                **      appropriately.  Routine exits when a key is found
1333                **      in buffer that isn't a scrolling key or when
1334                **      display timer times out.
1335                **
1336                ** History:
1337                **
1338                **    Date      Programmer            Modification
1339                **  --------    ----------    ----------------------------------
1340                **  10/19/82    B.S.          Updated documentation
1341                **  07/18/83    B.S.          Will not time out if a program
1342                **                            is running
1343                **
1344                ****************************************************************
1345                ****************************************************************
1346 0212E 716A =SCRLLR GOSUB  GETSTA          Save user status
1347 02132 853         ST=1   NoCur           Allow cursor out of display
1348 02135 8E16        GOSUBL BLDLCD          Build display
           7F
1349 0213B 8F00        GOSBVL =SETTM0         Start timeout timer
           000
1350 02142 7A3A  SCRL02 GOSUB  USRSTA         Need user status restored so that
1351                                          they will be re-saved by DSPUPD
1352                                          if it is called by CKSREQ
1353 02146 8E00        GOSUBL =SLEEP          Try to sleep
```

```
            00
1354 0214C 5F2          GONC   SCRL05          Key in buffer?  Yes, check it.
1355 0214F 765C         GOSUB  cksreq          No, see if anything has happened
1356 02153 870          ?ST=1  =PgmRun         Is a program running?
1357 02156 CE           GOYES  SCRL02          Yes, then don't allow time out
1358 02158 8F00         GOSBVL =ALMSRV         Check for timeout
            000
1359 0215F 863          ?ST=0  fTMOUT          If not time to turn off then
1360 02162 0E           GOYES  SCRL02          back to sleep.
1361 02164 3213         LC(3)  99*16+1         OFF key, 1 key in buffer
            6
1362 02169 1B34         DO=(5) =KEYPTR
            4F2
1363 02170 15C2         DATO=C 3
1364 02174 DA           A=C    A
1365 02176 F4           ASR    A               A(B)=OFF key
1366 02178 670A  USRSTj GOTO   USRSTA
1367              *-
1368              *-
1369 0217C 76E0  SCRL05 GOSUB  C=SCRL
1370 02180 B06          C=C+1  P               Is NEEDSC = F?
1371 02183 54F          GONC   USRSTj          No, then exit SCRLLR
1372 02186 1B44         DO=(5) =KEYBUF
            4F2
1373 0218D 749F         GOSUB  D1=FC
1374 02191 7B42         GOSUB  FINDDO          Read in A from DO
1375 02195 00           CON(2) =k#LFT          Cursor left
1376 02197 710          REL(3) SCRL10
1377 0219A 00           CON(2) =k#FLFT         Cursor far left
1378 0219C C30          REL(3) SCRL20
1379 0219F 00           CON(2) =k#RT           Cursor right
1380 021A1 740          REL(3) SCRL30
1381 021A4 00           CON(2) =k#FRT          Cursor far right
1382 021A6 770          REL(3) SCRL40
1383 021A9 00           CON(2) 0               Otherwise
1384 021AB 5CC          GONC   USRSTj
1385              *-
1386              *-
1387 021AE       SCRL10
1388 021AE 8E00         GOSUBL =POPBUF
            00
1389 021B4 7D6F  SCRL12 GOSUB  D1=FC
1390 021B8 14B          A=DAT1 B               Read FIRSTC
1391 021BB A6C          A=A-1  B               Decrement it
1392 021BE 4C0          GOC    SCRL13          Skip if it borrowed
1393 021C1 7DA0         GOSUB  SETFC           Set FIRSTC
1394 021C5 8EDC         GOSUBL BLDDSP
            6F
1395 021CB 8F00  SCRL13 GOSBVL =RPTKY          Check if key still down
            000
1396 021D2 511          GONC   SCRL22          Exit if not
1397 021D5 4ED          GOC    SCRL12          B.E.T.
1398              *-
1399              *-
1400 021D8 D0    SCRL20 A=0    A
```

```
1401 021DA 7490  SCRL21 GOSUB  SETFC          Set FIRSTC
1402 021DE 8E00         GOSUBL =POPBUF        Pop key out of buffer
           00
1403 021E4 694F  SCRL22 GOTO   SCRLLR         Loop back, wait for another key
1404            *-
1405            *-
1406 021E8       SCRL30
1407 021E8 8E00         GOSUBL =POPBUF
           00
1408 021EE 7C40  SCRL34 GOSUB  SCRL60         Point D1 past end of window
1409 021F2 31F5         LC(2)  95
1410 021F6 9E9          ?B>=C  B              Move past end of window?
1411 021F9 71           GOYES  SCRL35         Yes, then ignore key
1412 021FB 14F          C=DAT1 B              Read character
1413 021FE 96A          ?C=0   B              Is it a null?
1414 02201 F0           GOYES  SCRL35         Yes, then ignore key
1415 02203 B64          A=A+1  B
1416 02206 7860         GOSUB  SETFC          Set FIRSTC
1417 0220A 8E88         GOSUBL BLDDSP
           6F
1418 02210 8F00  SCRL35 GOSBVL =RPTKY
           000
1419 02217 5CC          GONC   SCRL22
1420 0221A 43D          GOC    SCRL34         (B.E.T.) No, then do scroll
1421            *-
1422            *-
1423 0221D 7D10  SCRL40 GOSUB  SCRL60         Point D1 past last char in window
1424 02221 31F5  SCRL45 LC(2)  95
1425 02225 9E9          ?B>=C  B              Would this be past end of buffer?
1426 02228 2B           GOYES  SCRL21         Yes, then done scrolling
1427 0222A 14F          C=DAT1 B              Read char
1428 0222D 96A          ?C=0   B              Is it a null?
1429 02230 AA           GOYES  SCRL21         Yes, then done scrolling
1430 02232 171          D1=D1+ 2              Move to next char
1431 02235 B64          A=A+1  B              Increment FIRSTC value
1432 02238 B65          B=B+1  B              Increment last char in window ptr
1433 0223B 55E          GONC   SCRL45         (B.E.T.)
1434            *-
1435            *-
1436 0223E 73EE  SCRL60 GOSUB  D1=FC
1437 02242 14B          A=DAT1 B              Read (FIRSTC)
1438 02245 1C8          D1=D1- (FIRSTC)-(WINDLN)
1439 02248 14F          C=DAT1 B              Read (WINDLN)
1440 0224B D1           B=0    A
1441 0224D AE8          B=A    B
1442 02250 A61          B=B+C  B              First char past end of window
1443 02253 3408         LC(5)  =DSPBFS
           4F2
1444 0225A C9           C=C+B  A
1445 0225C C9           C=C+B  A
1446 0225E 135          D1=C                  Point D1 at last char in window
1447 02261 171          D1=D1+ 2              Point past last char in window
1448 02264 03           RTNCC
1449            *-
1450            *-
```

```
1451 02266 1FA4 =C=SCRL D1=(5) =NEEDSC
           9F2
1452 0226D 14F          C=DAT1 B
1453 02270 01           RTN
1454            *-
1455            *-
1456 02272 7031  SETFC  GOSUB  SETFCA        Write out updated pointer
1457 02276 7629         GOSUB  RCLSTA
1458 0227A 841          ST=0   BitsOK
1459 0227D 6AF8         GOTO   PUTSTA        Restore user status
```

```
1460                    STITLE CRLFND,CRLFOF,CRLFSD  -  Send CR/LF
1461 02281 7E09 =XDELAY GOSUB  GETSTA
1462 02285 859          ST=1   XDelay
1463 02288 6FE8         GOTO   PUTSTA
1464            *-
1465            *-
```

```
1466                    EJECT
1467                ****************************************************************
1468                ****************************************************************
1469                **
1470                ** Name:(S) CRLFOF  -  Send cursor off/CR/LF to disp w/o delay
1471                ** Name:(S) CRLFND  -  Send CR/LF to display with no delay
1472                ** Name:(S) CRLFSD  -  Send CR/LF to display with delay
1473                **
1474                ** Category:   DSPUTL
1475                **
1476                ** Purpose:
1477                **      CRLFOF:  Send Cursor off, Replace Cursor, CR, LF to
1478                **               display with delay suppressed.
1479                **      CRLFND:  Send Replace Cursor, CR, LF to display with
1480                **               delay suppressed.
1481                **      CRLFSD:  Send Replace Cursor, CR, LF to display with
1482                **               delay.
1483                **
1484                ** Entry:
1485                **      P       =  0
1486                **
1487                ** Exit:
1488                **      P       =  0
1489                **
1490                ** Calls:       CRLFOF:  ESCSEQ, XDELAY, BF2DSP
1491                **              CRLFND:  XDELAY,BF2DSP
1492                **              CRLFSD:  BF2DSP
1493                **
1494                ** Uses.......
1495                **  Inclusive: A,B,C,D,D0,D1
1496                **
1497                ** Stk lvls:   5
1498                **
1499                ** History:
1500                **
1501                **    Date      Programmer              Modification
1502                **  --------    ----------   ----------------------------------
1503                **  11/01/83    B.S.         Added documentation
1504                **
1505                ****************************************************************
1506                ****************************************************************
1507 0228C B125 =RCCRLF NIBHEX B125DORORFF     Rep cur,CR,LF,End of string
         DORO
         FF
1508                *-
1509                *-
1510 02296 31C3 =CRLFOF LC(2)  \<\            Turn cursor off
1511 0229A 7321         GOSUB  ESCSEQ
1512 0229E 7FDF =CRLFND GOSUB  XDELAY         Suppress delay with CRLF
1513 022A2      =S-CRLF
1514 022A2 1FD4 =CRLFSD D1=(5) =DPOS
         9F2
1515 022A9 D2           C=0    A
1516 022AB 14D          DAT1=C B
1517 022AE 1FC8         D1=(5) =RCCRLF        Send Rep cur,CR,LF
```

```
                220
  1518 022B5 6859         GOTO   BF2DSP
```

```
1519                    STITLE SCNRT  -  Scan Right
1520                ********************************************************************
1521                ********************************************************************
1522                **
1523                ** Name:(S) SCNRT   -  Point Cursor Past Unprotected Field
1524                **
1525                ** Category:   DSPUTL
1526                **
1527                ** Purpose:
1528                **      Scans to right of cursor and returns A(A) pointing
1529                **      past end of unprotected field, a null byte or end
1530                **      of display buffer whichever comes first.
1531                **
1532                ** Entry:
1533                **      P      =  0
1534                **
1535                ** Exit:
1536                **      P      =  0
1537                **      A(A)=Points past unprotected display character
1538                **      Carry set if pointer points past DSPBFE (i.e. buffer
1539                **        is full and protected to end of buffer).
1540                **      B contains value of A at time of call.
1541                **      D(A) points past cursor position.
1542                **
1543                ** Calls:      CA=CUR,DO=CAA
1544                **
1545                ** Uses.......
1546                **  Exclusive:
1547                **  Inclusive: A,B,C,D(A),DO
1548                **
1549                ** Stk lvls:   1
1550                **
1551                ** History:
1552                **
1553                **    Date      Programmer              Modification
1554                **  --------   ----------   ---------------------------------
1555                **  10/19/82   B.S.         Updated documentation
1556                **  03/17/83   B.S.         Packed by calling subroutines
1557                **
1558                ********************************************************************
1559                ********************************************************************
1560 022B9 AF8    =SCNRT  B=A    W              Save A in B
1561 022BC 70ED        GOSUB  CA=CUR
1562 022C0 B64         A=A+1  B               Increment position
1563 022C3 70DD        GOSUB  DO=CAA          DO=C+A+A (Buffer pointer)
1564 022C7 134         DO=C                   DO=Buffer pointer
1565
1566 022CA D7          D=C    A               Save pointer past cursor position
1567 022CC 303         LCHEX  3               Mask for lower two bits
1568 022CF 0E02        C=C&A  P               Get lower 2 bits
1569 022D3 816         CSRC                   C(S)=Mod 4 of cursor position
1570 022D6 81C         ASRB
1571 022D9 81C         ASRB                   Divide cursor position by 4
1572 022DC 3475        LC(5)  (=DSPMSK)+23
          5F2
```

```
1573 022E3 E2          C=C-A  A          Calculate mask pointer
1574 022E5 136         CDOEX            DO=Mask ptr,C(A)=Buffer ptr
1575 022E8 1524        A=DATO S         Read mask nibble
1576 022EC 136         CDOEX            DO=Buffer ptr,C(A)=Mask ptr
1577 022EF 24          P=     4         Init bit count to 4 bits
1578 022F1 A4E  SCNR10 C=C-1  S         Is proper bit ready to shift off?
1579 022F4 4A0         GOC    SCNR20    Yes, then exit loop
1580 022F7 A44         A=A+A  S         No, then get next bit ready
1581 022FA 0D          P=P-1            Decrement count of bits
1582 022FC 54F         GONC   SCNR10    (B.E.T.)  Loop back
1583           *-
1584           *-
1585 022FF 132  SCNR20 ADOEX
1586 02302 8BE         ?A>=C  A         At end of display buffer?
1587 02305 53          GOYES  SCNR60    Yes, then RTNSC, P=0
1588 02307 132         ADOEX            No, check other exit conditions
1589 0230A 0D   SCNR30 P=P-1            Decr bit count.  All bits used?
1590 0230C 541         GONC   SCNR40    No, skip.
1591 0230F 136         CDOEX            Yes, then
1592 02312 180         DO=DO- 1         Decrement mask ptr
1593 02315 1524        A=DATO S         Read next mask nibble
1594 02319 136         CDOEX
1595 0231C 24          P=     4         Reset bit count
1596 0231E 50E         GONC   SCNR20    (B.E.T.) Check if still in buffer
1597           *-
1598           *-
1599 02321 14A  SCNR40 A=DATO B         Read buffer byte
1600 02324 968         ?A=0   B         Is it null?
1601 02327 E0          GOYES  SCNR50    Yes, then exit
1602 02329 161         DO=DO+ 2         Move buffer pointer
1603 0232C A44         A=A+A  S         No, then is byte protected?
1604 0232F 5AD         GONC   SCNR30    No, then loop back for next byte
1605 02332 181         DO=DO- 2         Undo last pointer move
1606 02335 132  SCNR50 ADOEX            Yes, then A(A)=buffer pointer
1607 02338 0C          P=P+1            Clear carry for normal return
1608 0233A 20   SCNR60 P=     0         Clear P before return
1609 0233C 01          RTN
```

```
1610                    STITLE MOVCUR  -  Move Cursor Right or Left
1611                ****************************************************************
1612                ****************************************************************
1613                **
1614                ** Name:    MOVCUR  -  Move Cursor Left or Right
1615                **
1616                ** Category:   LOCAL
1617                **
1618                ** Purpose:
1619                **      Moves cursor in specified direction skipping
1620                **      protected fields if necessary.
1621                **
1622                ** Entry:
1623                **      P       =  0
1624                **      CurLft(S0) set if cursor to move left, clear for
1625                **      move right
1626                **
1627                ** Exit:
1628                **      P       =  0
1629                **      CURSOR updated, P=0, Carry set if neither CURSOR
1630                **          nor FIRSTC move
1631                **
1632                ** Calls:      GETMSK
1633                **
1634                ** Uses.......
1635                **  Exclusive:
1636                **  Inclusive: A,B(B),C,D(A),D0
1637                **
1638                ** Stk lvls:   1
1639                **
1640                ** Detail:
1641                **      Trys to move cursor in specified direction.  If
1642                **      the cursor would land on a protected field then
1643                **      it is moved past that field.  If that protected
1644                **      field extends to end of buffer then FIRSTC is
1645                **      changed instead of CURSOR; this allows the protected
1646                **      part of the display to be viewed.
1647                **
1648                ** History:
1649                **
1650                **    Date      Programmer              Modification
1651                **  --------   ----------   ----------------------------------
1652                **  10/19/82   B.S.         Updated documentation
1653                **
1654                ****************************************************************
1655                ****************************************************************
1656 0233E 84B   =MOVCUR ST=0   NoChFC          Change FIRSTC if needed
1657 02341 1BE7    MOVC00 D0=(5) =CURSOR
           4F2
1658 02348 14E           C=DAT0 B
1659 0234B D7            D=C    A               Save original value in D(B)
1660 0234D 1AE7    MOVC10 D0=(4) =CURSOR
           4F
1661 02353 14A           A=DAT0 B
1662 02356 7850          GOSUB  MOVC80
```

```
1663 0235A 31F5          LC(2)  95
1664 0235E 9E6           ?A>C   B              Would be past end of display?
1665 02361 81            GOYES  MOVC40         Yes, then restore original value
1666 02363 148           DAT0=A B              No, then update cursor position
1667 02366 7058          GOSUB  GETMSK         Get bit map
1668 0236A 15A0          A=DAT0 1              Read mask nibble
1669 0236E 0E06          A=A&C  P
1670 02372 90C           ?A#0   P              Is it protected?
1671 02375 8D            GOYES  MOVC10         Yes, then keep looking
1672 02377 03            RTNCC
1673              *-
1674              *-
1675 02379 DB    MOVC40  C=D    A              C(B)=Original cursor
1676 0237B 14C           DAT0=C B              Restore original cursor
1677 0237E 87B           ?ST=1  NoChFC         Okay to change FIRSTC?
1678 02381 00            RTNYES                No, then return with carry set
1679 02383 181           D0=D0- 2              Point at FIRSTC
1680 02386 14A           A=DAT0 B              Read FIRSTC
1681 02389 7520          GOSUB  MOVC80
1682 0238D 853           ST=1   NoCur
1683 02390 31F5          LC(2)  95
1684 02394 AE5           B=C    B
1685 02397 1937          D0=(2) =WINDLN
1686 0239B 14E           C=DAT0 B              Read window length
1687 0239E B61           B=B-C  B              B(B) is last legal FIRSTC
1688 023A1 9E0           ?A>B   B              Is it in a legal range?
1689 023A4 00            RTNYES                No, return without changing it
1690 023A6 1BC7  SETFCA  D0=(5) =FIRSTC
           4F2
1691 023AD 148           DAT0=A B              Change FIRSTC
1692 023B0 03            RTNCC
1693              *-
1694              *-
1695 023B2 860   MOVC80  ?ST=0  CurLft         Moving cursor left?
1696 023B5 70            GOYES  MOVC90         No, then skip
1697 023B7 A6C           A=A-1  B              Yes, then decrement value
1698 023BA 01            RTN
1699 023BC B64   MOVC90  A=A+1  B              Increment value
1700 023BF 01            RTN
1701              *-
1702              *-
```

```
1703                    STITLE ESCSEQ
1704                ********************************************************************
1705                ********************************************************************
1706                **
1707                ** Name:(S) ESCSEQ  -  Send Escape Sequence to Display
1708                **
1709                ** Category:   DSPUTL
1710                **
1711                ** Purpose:
1712                **      Sends an escape to display followed by a specified
1713                **      character.
1714                **
1715                ** Entry:
1716                **      P      = 0
1717                **      C(B)=Character to follow escape character.
1718                **
1719                ** Exit:
1720                **      P      = 0
1721                **
1722                ** Calls:      DSPCHC
1723                **
1724                ** Uses.......
1725                **  Exclusive: C(B)
1726                **  Inclusive: A(W),B(W),C(W),D(W),D0,D1,RAM(See DSPCHA)
1727                **
1728                ** Stk lvls:   4
1729                **
1730                ** History:
1731                **
1732                **    Date     Programmer              Modification
1733                **  --------   ----------   ---------------------------------
1734                **  07/15/82   B.S.         Added documentation
1735                **
1736                ********************************************************************
1737                ********************************************************************
1738 023C1 06    =ESCSEQ RSTK=C                  Save character
1739 023C3 31B1         LCHEX  1B                Constant ASCII Esc
1740 023C7 7178         GOSUB  DSPCHC
1741 023CB 07           C=RSTK                   Restore character
1742 023CD 6E68         GOTO   DSPCHC            Send character & return
```

```
1743                    STITLE FINDA  -  Find A(B) From Table
1744            ******************************************************************
1745            ******************************************************************
1746            **
1747            ** Name:(S) FINDA   -  Look For A(B) In A Table And Jump
1748            ** Name:(S) FINDDO  -  Look For (DO) In A Table And Jump
1749            **
1750            ** Category:   GENUTL
1751            **
1752            ** Purpose:
1753            **      Searches a table following GOSUB for a byte matching
1754            **      A[B] and jumps to address specified for that value.
1755            **
1756            ** Entry:
1757            **      FINDA:
1758            **        A(B)=byte to be found
1759            **      FINDDO:
1760            **        (DO)=byte to be found
1761            **      Table of bytes and address offsets must follow GOSUB
1762            **      The call should look as follows:
1763            **
1764            **           GOSBVL =FINDA      <---GOSUB is followed by table
1765            **           CON(2) \Q\         <---Byte to be matched
1766            **           REL(3) ESCQ        <---Where to jump if matched
1767            **           CON(2) \R\
1768            **           REL(3) ESCR
1769            **           CON(2) \C\
1770            **           REL(3) ESCC
1771            **           .
1772            **           .
1773            **           .
1774            **           CON(2) 0           <---Null byte terminates table
1775            **                              <---Followed by code to execute
1776            **                                  if no match is found
1777            **      Entry points:
1778            **        1) FNDDO+ - Increments DO 1 byte, then reads in A(B)
1779            **        2) FINDDO - Reads in A(B) from DO
1780            **        3) FINDA  - Assumes byte to compare already in A(B)
1781            **
1782            ** Exit:
1783            **      P      =  0
1784            **
1785            ** Calls:      None
1786            **
1787            ** Uses.......
1788            **  Inclusive: C(A)
1789            **
1790            ** Stk lvls:   0
1791            **
1792            ** Detail:
1793            **      This routine uses 3 nibble self-relative offsets
1794            **
1795            ** Algorithm:
1796            **      Pops address off return stack and uses that address
1797            **      as the start of a table of alternating byte to be
```

```
1798                **      compared and 3-nibble relative offsets of where to
1799                **      jump if that byte matchs what is in A(B).  The last
1800                **      entry in the table should be a 0 byte followed by
1801                **      the code to execute if no match is found.
1802                **
1803                ** History:
1804                **
1805                **    Date     Programmer            Modification
1806                **   --------  ----------  ----------------------------------
1807                **  09/13/82   B.S.        Wrote routine to replace BYTSCN
1808                **  09/14/82   B.S.        Changed to fall thru to otherwise
1809                **                         code
1810                **
1811                ********************************************************************
1812                ********************************************************************
1813 023D1 07     FINDOO C=RSTK
1814 023D3 161         DO=DO+ 2
1815 023D6 136         CDOEX                  Restore DO, C(A)=return address
1816 023D9 6640        GOTO   FIND50
1817                *-
1818                *-
1819 023DD 161  =FNDDO+ DO=DO+ 2
1820 023E0 14A  =FINDDO A=DATO B
1821 023E3 07   =FINDA  C=RSTK                 Pop address from return stack
1822 023E5 136         CDOEX                   Put address in DO
1823 023E8 06          RSTK=C                  Save old DO on RSTK
1824 023EA 14E   FIND10 C=DATO B               Read a byte
1825 023ED 96A         ?C=0   B                At end of list?
1826 023F0 1E          GOYES  FINDOO           Yes, then exit loop
1827 023F2 164         DO=DO+ 5
1828 023F5 966         ?A#C   B
1829 023F8 2F          GOYES  FIND10
1830 023FA 182         DO=DO- 3                Back up to jump offset
1831                *       Now fall into REL3DO to do offset jump
```

```
1832                    STITLE REL3DO  -  3-Nibble-Relative at D0 Jump
1833                ****************************************************************
1834                ****************************************************************
1835                **
1836                ** Name:    REL3D0  -  Rel Jump to 3 Nibble Offset at D0
1837                **
1838                ** Category:   GENUTL
1839                **
1840                ** Purpose:
1841                **      Read a 3 nibble relative address at D0 and jumps to it
1842                **
1843                ** Entry:
1844                **      D0 points to 3 nibble self-relative offset to jump to
1845                **      RSTK=Value to restore to D0 before jumping to address
1846                **
1847                ** Exit:
1848                **      P      =  0
1849                **      D0=value that had been on RSTK
1850                **
1851                ** Calls:      None
1852                **
1853                ** Uses.......
1854                **  Inclusive: C(A)
1855                **
1856                ** Stk lvls:   -1 (pops value off return stack)
1857                **
1858                ** History:
1859                **
1860                **    Date     Programmer              Modification
1861                **  --------   ----------   ----------------------------------
1862                **  10/14/82   B.S.         Created entry point
1863                **
1864                ****************************************************************
1865                ****************************************************************
1866                *       Note:  Above code falls into here
1867 023FD 23    =REL3D0 P=     3
1868 023FF 3100         LCHEX  00
1869 02403 1563         C=DAT0 X              Read offset
1870 02407 0B           CSTEX                 Swap it into status bits
1871 02409 86B          ?ST=0  11             Is is a negative offset
1872 0240C 60           GOYES  FIND40         No, then okay
1873 0240E 31FF         LCHEX  FF             Yes, then do sign extend
1874 02412 0B     FIND40 CSTEX                Swap it back into C(X)
1875 02414 132          ADOEX                 A(A)=Table pointer,D0=Old A
1876 02417 CA           A=A+C  A              Add offset
1877 02419 07           C=RSTK                C=Original D0
1878 0241B DE           ACEX   A              A(A)=Orig D0, C(A)=Table Pointer
1879 0241D 132          ADOEX                 A(A)=Old A, D0=Original D0
1880 02420 20     FIND50 P=     0             Must leave P=0
1881 02422 06           RSTK=C                Push address on stack
1882 02424 03           RTNCC                 Jump to address specified
1883                *-
1884                *-
```

```
1885                    STITLE TBLJMP,TBLJMC  -  Table Jump
1886                ****************************************************************
1887                ****************************************************************
1888                **
1889                ** Name:(S) TBLJMP  -  Indexed table jump
1890                ** Name:(S) TBLJMC  -  Indexed table jump
1891                **
1892                ** Category:   GENUTL
1893                **
1894                ** Purpose:
1895                **      Performs an indexed table jump into a table of 3-nibble
1896                **      relative offets following GOSUB.
1897                **
1898                ** Entry:
1899                **      Table of relative offsets must follow GOSUB
1900                **      TBLJMP:  P = index of table to jump to
1901                **      TBLJMC:  C(0) = index of table to jump to
1902                **
1903                ** Exit:
1904                **      P      =  0
1905                **
1906                ** Calls:      None
1907                **
1908                ** Uses.......
1909                **  Inclusive: C(A)
1910                **
1911                ** Stk lvls:   0
1912                **
1913                ** Detail:
1914                **      Pops address off stack and adds 3 times the index to
1915                **      it.  It then uses REL3DO to jump to the address
1916                **      specified by that table entry.
1917                **
1918                ** History:
1919                **
1920                **    Date     Programmer             Modification
1921                **  --------   ----------   ---------------------------------
1922                **  10/14/82   B.S.         Created routine to replace CASE.
1923                **
1924                ****************************************************************
1925                ****************************************************************
1926 02426 80D0 =TBLJMC P=C    0              Move pointer to P
1927 0242A 07   =TBLJMP C=RSTK                Pop table address from stack
1928 0242C 0D           P=P-1
1929 0242E 4B0          GOC    TBLJM1
1930 02431 809          C+P+1
1931 02434 809          C+P+1
1932 02437 809          C+P+1
1933 0243A 136   TBLJM1 CDOEX                 C(A)=Old DO, DO=ptr to offset
1934 0243D 06           RSTK=C                Save old DO value on stack
1935 0243F 6DBF         GOTO   REL3DO         Now jump do offset jump
```

```
1936                    STITLE DSPRST  -  Display Reset
1937                ********************************************************************
1938                ********************************************************************
1939                **
1940                ** Name:(S) DSPRST  -  Display reset
1941                **
1942                ** Category:   DSPUTL
1943                **
1944                ** Purpose:
1945                **      Resets display driver pseudo-device:  clears buffer,
1946                **      display mask, cursor position, first character,
1947                **      status, and window.
1948                **
1949                ** Entry:
1950                **
1951                ** Exit:
1952                **      P      =  0
1953                **
1954                ** Calls:      None
1955                **
1956                ** Uses.......
1957                **  Inclusive: C(W),P,D0
1958                **
1959                ** Stk lvls:   0
1960                **
1961                ** History:
1962                **
1963                **    Date     Programmer             Modification
1964                **  --------   ----------   ----------------------------------
1965                **  10/25/83   B.S.         Added documentation
1966                **
1967                ********************************************************************
1968                ********************************************************************
1969 02443 1B87 =DSPRST D0=(5) (=DSPSTA)+3    Will clear to end of DSPMSK
           4F2
1970 0244A AF2          C=0    W
1971 0244D 22           P=     2             Clear 4+4+96*2+24 nibbles
1972 0244F 1547  DSPRS1 DAT0=C W
1973 02453 16F          D0=D0+ 16
1974 02456 0C           P=P+1
1975 02458 56F          GONC   DSPRS1        Loop until done
1976 0245B 1A17         D0=(4) =WINDST
           4F
1977 02461 3300         LC(4)  21~0          WINDST=0,WINDLN=21
           51
1978 02467 15C3         DAT0=C 4
1979 0246B 03           RTNCC                Return (P=0,Carry clear)
1980                *-
1981                *-
```

```
1982                    STITLE NOKEY+
1983                ********************************************************************
1984                ********************************************************************
1985                **
1986                ** Name:    NOKEY+  -  Flush Key Buffer Except ATTN Key
1987                **
1988                ** Category:   KEYUTL
1989                **
1990                ** Purpose:
1991                **      Flushes key buffer except leaves ATTN key if present
1992                **
1993                ** Entry:
1994                **      P       =  0
1995                **
1996                ** Exit:
1997                **      P       =  0
1998                **
1999                ** Calls:      NOKEYS
2000                **
2001                ** Uses.......
2002                **  Exclusive: A(S)
2003                **  Inclusive: A(B),C(B),A(S)
2004                **
2005                ** Stk lvls:   0
2006                **
2007                ** History:
2008                **
2009                **    Date      Programmer              Modification
2010                **  --------    ----------   ----------------------------------
2011                **  05/19/83    B.S.         Created routine
2012                **
2013                ********************************************************************
2014                ********************************************************************
2015 0246D 1B24 =NOKEY+ DO=(5) =ATNFLG
           4F2
2016 02474 1524         A=DAT0 S
2017 02478 94C          ?A#0   S
2018 0247B 00           RTNYES
2019 0247D 8D00 =nokeys GOVLNG =NOKEYS
           000
2020 02484              END
```

```
=A=CUR   Abs    8362 #020AA -  1218   851  1275
 ADDC05  Abs    7855 #01EAF -  1013  1005
 ADDC10  Abs    7878 #01EC6 -  1020  1037
 ADDC15  Abs    7905 #01EE1 -  1028  1023
 ADDC17  Abs    7918 #01EEE -  1032   998
 ADDC20  Abs    7922 #01EF2 -  1035   996
 ADDC25  Abs    7925 #01EF5 -  1036  1009
 ADDCHR  Abs    7782 #01E66 -   988
 ALMSRV  Ext                -  1358
 ANNAD1  Abs  188672 #2E100 -    50   145
 ANNAD4  Abs  189262 #2E34E -    50   176
 ASCII   Ext                -   297
 ATNFLG  Abs  193602 #2F442 -    50   942  2015
=AVM2DS  Abs    7169 #01C01 -   627
 AVMEMS  Abs  193940 #2F594 -    50   627
 AtEnd   Abs       0 #00000 -    22  1179  1183  1192
=BF2BLD  Abs    6292 #01894 -   105
=BF2DPP  Abs    7158 #01BF6 -   625
=BF2DS+  Abs    7176 #01C08 -   628
=BF2DSP  Abs    7182 #01C0E -   630   105   626   641  1518
 BLD003  Abs    6350 #018CE -   126   117   860
 BLD004  Abs    6353 #018D1 -   127   110   122   125
 BLD005  Abs    6403 #01903 -   145   135   143
 BLD006  Abs    6426 #0191A -   152   150
 BLD020  Abs    6467 #01943 -   166   163
 BLD080  Abs    6535 #01987 -   189   182   185
=BLDB10  Abs    6591 #019BF -   277   334
 BLDB20  Abs    6624 #019E0 -   287   282   285
 BLDB30  Abs    6654 #019FE -   299   295
 BLDB40  Abs    6662 #01A06 -   302   489
 BLDB45  Abs    6730 #01A4A -   323   325
 BLDB50  Abs    6742 #01A56 -   327   317
 BLDB60  Abs    6755 #01A63 -   332   329
=BLDBIT  Abs    6588 #019BC -   276   175   479
 BLDDS*  Abs    6305 #018A1 -   109   869  1143
=BLDDSP  Abs    6296 #01898 -   106   956  1394  1417
=BLDLCD  Abs    6300 #0189C -   107   375   836   865  1348
 BLDLCj  Abs    6870 #01AD6 -   375   455
=BitsOK  Abs       1 #00001 -    24   107   193   454   823   895   994  1083
                                1266  1458
=C=SCRL  Abs    8806 #02266 -  1451  1369
 CA=CUR  Abs    8352 #020A0 -  1216  1209  1561
 CHRB10  Abs    6584 #019B8 -   274   269
=CHRBUF  Abs    6559 #0199F -   266   287
 CKSREQ  Ext                -   915
 CLEAR.  Abs    8441 #020F9 -  1278  1283
 CLEAR0  Abs    8405 #020D5 -  1266  1294
 CLEAR1  Abs    8411 #020DB -  1268  1271
 CLEAR2  Abs    8456 #02108 -  1286  1262
 CLEAR3  Abs    8461 #0210D -  1288  1273
 CLEARD  Abs    8386 #020C2 -  1261   828   988  1095  1105  1116  1126  1131
                                1152  1174  1178  1204  1256
=CRLFND  Abs    8862 #0229E -  1512
=CRLFOF  Abs    8854 #02296 -  1510
=CRLFSD  Abs    8866 #022A2 -  1514
```

```
 CURSOR  Abs  193662 #2F47E -    50   562   811   859  1218  1657  1660
=Clear   Abs       5 #00005 -    28   931  1172  1261  1263
 CurLft  Abs       0 #00000 -    21  1028  1100  1110  1117  1127  1695
=CurOff  Abs       6 #00006 -    29   452   834   842   935  1022  1138  1146
                                1171
 D-E.C1  Abs    8081 #01F91 -  1106  1112
 D-E.D1  Abs    8119 #01FB7 -  1120  1121
 D-ES.C  Abs    8077 #01F8D -  1105  1075
 D-ES.D  Abs    8106 #01FAA -  1116  1077
 D-ESCX  Abs    8328 #02088 -  1204  1073
 D-ESC<  Abs    8179 #01FF3 -  1146  1065
 D-ESC>  Abs    8156 #01FDC -  1138  1063
 D-ESCC  Abs    8050 #01F72 -  1095  1053
 D-ESCD  Abs    8129 #01FC1 -  1126  1055
 D-ESCE  Abs    8236 #0202C -  1170  1067
 D-ESCH  Abs    8139 #01FCB -  1131  1057
 D-ESCJ  Abs    8188 #01FFC -  1151  1059
 D-ESCK  Abs    8188 #01FFC -  1152  1061
 D-ESCP  Abs    8256 #02040 -  1178  1069  1071
 D-ESCQ  Abs    7932 #01EFC -  1040  1047  1049
 D-ESCR  Abs    8035 #01F63 -  1087  1051
 DO=CAA  Abs    8343 #02097 -  1210  1563
=DO=CUR  Abs    8339 #02093 -  1209   814  1036  1096  1106
 DO=FC   Abs    8041 #01F69 -  1091   111   132   154  1133  1289
 D1=FC   Abs    8485 #02125 -  1297   456  1373  1389  1436
 DD1CTL  Abs  189439 #2E3FF -    50   190   362   365   495
 DD3ST   Abs  188676 #2E104 -    50   312
 DELAYT  Abs  194888 #2F948 -    50   938
=DOSCRL  Abs    7391 #01CDF -   839
 DPOS    Abs  194893 #2F94D -    50  1514
 DSP-BS  Abs    8129 #01FC1 -  1125   986
 DSP-CR  Abs    7372 #01CCC -   828   980
 DSP-EC  Abs    7738 #01E3A -   971   984
 DSP-LF  Abs    7611 #01DBB -   928   982
 DSPBFS  Abs  193664 #2F480 -    50   136   172   476   887  1217  1265  1443
 DSPC33  Abs    7466 #01D2A -   865   841   843
 DSPC35  Abs    7470 #01D2E -   868   864
 DSPCH*  Abs    7282 #01C72 -   794   783
 DSPCH.  Abs    7248 #01C50 -   782   776
 DSPCH@  Abs    7280 #01C70 -   793   788
=DSPCHA  Abs    7230 #01C3E -   774   637
=DSPCHC  Abs    7228 #01C3C -   773  1740  1742
 DSPCHX  Abs  194164 #2F674 -    50   785
=DSPCL?  Abs    8374 #020B6 -  1255
 DSPCR1  Abs    7395 #01CE3 -   840   835
 DSPCR2  Abs    7453 #01D1D -   859   857
 DSPCR3  Abs    7460 #01D24 -   863   848
 DSPCR4  Abs    7474 #01D32 -   869   904
 DSPCR5  Abs    7560 #01D88 -   900   909
 DSPCR7  Abs    7589 #01DA5 -   912   886   893   960
=DSPDLY  Abs    7599 #01DAF -   921
 DSPLF0  Abs    7634 #01DD2 -   935   922
 DSPLF1  Abs    7688 #01E08 -   951
 DSPLF2  Abs    7694 #01E0E -   955   949
 DSPLF3  Abs    7702 #01E16 -   957   964
```

```
 DSPMSK  Abs  193856 #2F540 -    50   559  1292  1572
 DSPRS1  Abs    9295 #0244F -  1972  1975
=DSPRST  Abs    9283 #02443 -  1969
 DSPS-0  Abs    7756 #01E4C -   978   799
 DSPS-1  Abs    7939 #01F03 -  1045   800
 DSPS-2  Abs    7317 #01C95 -   808   801
 DSPS-3  Abs    8025 #01F59 -  1081   802
 DSPS20  Abs    7314 #01C92 -   807   810
 DSPS21  Abs    7342 #01CAE -   816   822
 DSPS22  Abs    7362 #01CC2 -   823   818
 DSPSTA  Abs  193653 #2F475 -    50   502   510   513   518  1969
 DSPTIM  Abs    7009 #01B61 -   493   899   955
 DSPU01  Abs    6886 #01AE6 -   452   196
 DSPU02  Abs    6891 #01AEB -   454   925
 DSPU2j  Abs    7607 #01DB7 -   925   936
 DSPU40  Abs    6976 #01B40 -   483   474
 DSPU50  Abs    6994 #01B52 -   488   486
 DSPU60  Abs    7000 #01B58 -   490   480
=DSPUP1  Abs    6878 #01ADE -   448
=DSPUPD  Abs    6874 #01ADA -   447
 ESC>10  Abs    8162 #01FE2 -  1140  1148
 ESCD10  Abs    8069 #01F85 -  1101  1128
 ESCK10  Abs    8216 #02018 -  1161  1156  1159  1167
 ESCP10  Abs    8273 #02051 -  1184  1182
 ESCP20  Abs    8315 #0207B -  1198  1193  1196
 ESCS0j  Abs    8073 #01F89 -  1102  1109  1113  1122
=ESCSEQ  Abs    9153 #023C1 -  1738  1511
 ESCSET  Abs    7741 #01E3D -   972   825  1084  1206
 ESCST0  Abs    8025 #01F59 -  1080  1041  1088  1099  1102  1175  1201
 ESCSTA  Abs  193659 #2F47B -    50   796   811   972  1140
 ESCSTj  Abs    8252 #0203C -  1175  1135  1165
 EscSt0  Abs       0 #00000 -    42  1082
 EscSt1  Abs       1 #00001 -    43   971
 EscSt2  Abs       2 #00002 -    44  1205
 EscSt3  Abs       3 #00003 -    45   824
 FIND00  Abs    9169 #023D1 -  1813  1826
 FIND10  Abs    9194 #023EA -  1824  1829
 FIND40  Abs    9234 #02412 -  1874  1872
 FIND50  Abs    9248 #02420 -  1880  1816
=FINDA   Abs    9187 #023E3 -  1821   978  1045
=FINDD0  Abs    9184 #023E0 -  1820  1374
 FIRSTC  Abs  193660 #2F47C -    50   120   159   859   878  1091  1297  1438
                                1690
=FNDD0+  Abs    9181 #023DD -  1819
=GETMSK  Abs    7098 #01BBA -   559  1024  1667
=GETSTA  Abs    7059 #01B93 -   510   106   447   782   839   921  1255  1346
                                1461
 IOFND0  Ext                -   284
=Insert  Abs       7 #00007 -    30   485   995  1040  1087  1170
 KEYBUF  Abs  193604 #2F444 -    50  1372
 KEYPTR  Abs  193603 #2F443 -    50  1362
 LCDTAB  Abs    6764 #01A6C -   337   306
 LeftA   Abs       3 #00003 -    14   148   151
 MOVC00  Abs    9025 #02341 -  1657  1030
 MOVC10  Abs    9037 #0234D -  1660  1671
```

```
 MOVC40  Abs    9081 #02379 -  1675  1665
 MOVC80  Abs    9138 #023B2 -  1695  1662  1681
 MOVC90  Abs    9148 #023BC -  1699  1696
=MOVCUR  Abs    9022 #0233E -  1656  1101  1111  1120
 MOveu3  Ext                -  1190
 Moved3  Ext                -  1017
 NEEDSC  Abs  194890 #2F94A -    50   991  1451
=NOKEY+  Abs    9325 #0246D -  2015   868   946
 NOKEYS  Ext                -  2019
 NOTIM-  Abs    6863 #01ACF -   371   453
 NOTIM1  Abs    6850 #01AC2 -   367   370
=NOTIME  Abs    6830 #01AAE -   362   450   463   467   968
 NOTIMX  Abs    7731 #01E33 -   967   934
 NOTIMj  Abs    7734 #01E36 -   968   941   945
 NoChFC  Abs      11 #0000B -    34  1029  1656  1677
 NoCur   Abs       3 #00003 -    26   109   794   833   844   932  1031  1139
                              1147  1347  1682
 POPBUF  Ext                -  1388  1402  1407
 PROMPT  Ext                -   625
 PUTST1  Abs    7049 #01B89 -   504   514
 PUTSTA  Abs    7032 #01B78 -   499   102   372   950   974  1032  1257  1459
                              1463
 PUTSTj  Abs    6288 #01890 -   102   108
 PUTSj2  Abs    7752 #01E48 -   974   990
 PgmRun  Ext                -  1356
 Phase   Abs       2 #00002 -    25   194   473   475   483
 R<RSTK  Ext                -   775
 RArrow  Abs       0 #00000 -    20   127   165   181
=RCCRLF  Abs    8844 #0228C -  1507  1517
=RCLSTA  Abs    7072 #01BA0 -   513   875   902   930   958  1457
=REL3D0  Abs    9213 #023FD -  1867  1935
 REPCUR  Ext                -   484
 RPTKY   Ext                -  1395  1418
 RSTK<R  Ext                -   778
=ResetC  Abs       8 #00008 -    31   829  1173  1272  1286  1288
 RightA  Abs       4 #00004 -    15   179   187
=S-CRLF  Abs    8866 #022A2 -  1513
 SCNR10  Abs    8945 #022F1 -  1578  1582
 SCNR20  Abs    8959 #022FF -  1585  1579  1596
 SCNR30  Abs    8970 #0230A -  1589  1604
 SCNR40  Abs    8993 #02321 -  1599  1590
 SCNR50  Abs    9013 #02335 -  1606  1601
 SCNR60  Abs    9018 #0233A -  1608  1587
=SCNRT   Abs    8889 #022B9 -  1560   997  1153  1180
 SCRL02  Abs    8514 #02142 -  1350  1357  1360
 SCRL05  Abs    8572 #0217C -  1369  1354
 SCRL10  Abs    8622 #021AE -  1387  1376
 SCRL12  Abs    8628 #021B4 -  1389  1397
 SCRL13  Abs    8651 #021CB -  1395  1392
 SCRL20  Abs    8664 #021D8 -  1400  1378
 SCRL21  Abs    8666 #021DA -  1401  1426  1429
 SCRL22  Abs    8676 #021E4 -  1403  1396  1419
 SCRL30  Abs    8680 #021E8 -  1406  1380
 SCRL34  Abs    8686 #021EE -  1408  1420
 SCRL35  Abs    8720 #02210 -  1418  1411  1414
```

```
 SCRL40  Abs    8733 #0221D -  1423  1382
 SCRL45  Abs    8737 #02221 -  1424  1433
 SCRL60  Abs    8766 #0223E -  1436  1408  1423
=SCRLLR  Abs    8494 #0212E -  1346   951  1403
 SCROLT  Abs  194886 #2F946 -    50   845   897
 SETFC   Abs    8818 #02272 -  1456  1393  1401  1416
 SETFCA  Abs    9126 #023A6 -  1690   894  1119  1456
 SETSTA  Abs    7085 #01BAD -   518   500
 SETTMO  Ext                 -  1349
 SLEEP   Ext                 -   906   962  1353
=STOSTA  Abs    7083 #01BAB -   517
 SYSFLG  Abs  194265 #2F6D9 -    50   677
=SavLvl  Abs       5 #00005 -    49   774   777
 TBLJM1  Abs    9274 #0243A -  1933  1929
=TBLJMC  Abs    9254 #02426 -  1926   798
=TBLJMP  Abs    9258 #0242A -  1927
 TIMER1  Abs  189432 #2E3F8 -    50   365   495
 TimOut  Abs      10 #0000A -    33   448   498   903   959
=USRSTA  Abs    7040 #01B80 -   502   905   912   928   961  1350  1366
 USRSTj  Abs    8568 #02178 -  1366  1371  1384
 UpdOff  Abs       4 #00004 -    27   371   449   490
 WINDLN  Abs  193651 #2F473 -    50   118   159   464   468   849   876   878
                             1438  1685
 WINDST  Abs  193649 #2F471 -    50   468  1976
 WRTTM1  Ext                 -   494
=XDELAY  Abs    8833 #02281 -  1461  1512
=XDelay  Abs       9 #00009 -    32   840   933   967  1462
 bALTCH  Ext                 -   283
 cksreq  Abs    7593 #01DA9 -   915   901   929   957  1355
=fCALC?  Abs    7214 #01C2E -   677   131
 fTMOUT  Abs       3 #00003 -    38  1359
 k#FLFT  Ext                 -  1377
 k#FRT   Ext                 -  1381
 k#LFT   Ext                 -  1375
 k#RT    Ext                 -  1379
=nokeys  Abs    9341 #0247D -  2019
```

Input Parameters

  Source file name is SB&DSP::MS

  Listing file name is SB/DSP:TI:ML::-1

  Object file name is SB%DSP:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    M   M  BBBB    &     RRRR   00000  M   M
 2            *    MM MM  B   B  & &    R   R  0   0  MM MM
 3            *    M M M  B   B  & &    R   R  0   0  M M M
 4            *    M M M  BBBB    &     RRRR   0   0  M M M
 5            *    M   M  B   B  & & &  R R    0   0  M   M
 6            *    M   M  B   B  &  &   R  R   0   0  M   M
 7            *    M   M  BBBB    && &  R   R  00000  M   M
 8
 9                   TITLE  HP-71 Rom Test <831212.1205>
10 02484            ABS    #2484
11            *
12            ********************************************************************
13            ********************************************************************
14            **
15            ** Name:    ROMTST  -  System ROM checksum test
16            **
17            ** Category:   SYSTEM
18            **
19            ** Purpose:
20            **      Perform ROM checksum on the four system ROMs.
21            **
22            ** Entry:
23            **      P      =  0
24            **
25            ** Exit:
26            **      P=0, carry set
27            **
28            ** Calls:      BF2BLD, DSPCHA, BLDDSP, RCKBp
29            **
30            ** Uses.......
31            **  Exclusive: A,B,C,D,R0,R1,D1,D0,P, OUT register
32            **  Inclusive: A,B,C,D,R0,R1,D1,D0,P, OUT register
33            **
34            ** Stk lvls:   3
35            **
36            ** Detail:
37            **   The ROM test is performed as a checksum, on all 4 chips.
38            **   Results are reported on the beeper as follows:
39            **     Three tones are used (at nominal 400Khz) ---
40            **       617Hz Medium tone indicates "results follow"
41            **       775Hz High tone indicates "ROM chip OK"; chip #1 will
42            **                 beep once, chip #2 will beep twice, etc.
43            **       438Hz Low tone indicates "ROM chip BAD"; chip #1 will
44            **                 beep once, chip #2 will beep twice, etc.
45            **
46            **-----------------------------------------------------------
47            **
48            **   This ROM test depends on system RAM (namely the RAM
49            **   accessed by the display code).  A RAM failure might
50            **   prevent the ROM test from being run properly, but this
51            **   would most likely show up as a complete failure of
52            **   the HP-71: a permanent "Memory Lost" message, or the
53            **   display locked up with other characters.
54            **
55            **   The ROM test does not depend on code in ROM chips 2,3
```

```
                **   and 4.  A failure of ROM chip #1 might preclude a test
                **   of the other three ROMs.
                **
                **   The ROM test includes a communication link with an
                **   Electronic Tester device.  This link was set up for
                **   automatic testing of production line components, and
                **   implemented on the Self-Check ROM.  The ET communication
                **   scheme is as follows:
                **     -- When the test is completed, a preamble is sent out
                **        on the OUT register indicating "test results follow":
                **
                **            OUT regisiter= "OFF" , held for 600 usec (nominal
                **              at 400KHZ)
                **
                **     -- After preamble, report test results on OUT register
                **        as follows:  OUT reg= "0xt", where
                **            "t"= test code ("t"=8 for ROM test)
                **        and "x"=bitwise results (e.g., "x"="1010" means chips
                **              one and three are bad, chips 2 and 4 are good).
                **        This is held on the OUT register for 600 usec, also.
                **
                **    If the ET is not hooked up, no problem: this is a one-way
                **    communication, at this point.  See the HP-71 Self-Check
                **    ROM for more details on the ET communication link; the
                **    full scheme includes two-way communications.
                **
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------   ----------   ----------------------------------
                **  03/26/83   MB           Wrote ROM test
                **
                ********************************************************************
                ********************************************************************
                *
02484           =ROMTST
02484            ROMCHK
02484 1F48 520          D1=(5) =RmTsMs       Point to ROM TEST message.
0248B 8E00 00           GOSUBL =BF2BLD       To display, build display.
02491 AF0               A=0    W
02494 100               R0=A                 R0(S)= beep counter (temporary),
                *                               R0(XS)= error report flags.
                *                            R1(S)= chip counter,
                *                               R1(A)= chip address.
                *
02497 B44        CHKSUM A=A+1  S             Count chips.
0249A 101               R1=A                 Save counter.
0249D 303               LCHEX  3             For ASCII code.
024A0 DA                A=C    A             A(0)= 3.
024A2 810               ASLC                 A(B)= ASCII code for chip#.
024A5 E6                C=C+1  A             C(0)= 4 = max chip count.
024A7 98A               ?A<=C  P             More chips to check?
```

```
    109 024AA F1          GOYES  CHKSM1          Yes.
    110             *
    111             *--- Exit ROM test with ET communications.
    112 024AC D2          C=0    A               Set C(X)= OFF
    113 024AE CE          C=C-1  A                 for ET's preamble.
    114 024B0 7700        GOSUB  ROMC19          Preamble, 617 usec wait.
    115             *
    116 024B4 118         C=R0                   Set C(X)= 0x8, where
    117 024B7 3108        LCHEX  80                x= bad chip flags.
    118             *                    Fall into ROMC19, send out 0x8.
    119 024BB BB6  ROMC19 CSR    X
    120 024BE 801         OUT=C
    121             *
    122 024C1 B26  ROMC21 C=C+1  XS              617 usec wait (400KHZ nominal).
    123 024C4 5CF         GONC   ROMC21
    124 024C7 01          RTN                <----<<<  EXIT.
    125             *
    126             *---
    127 024C9 7BA0 CHKSM1 GOSUB  DspBld          Display chip #, build dsp.
    128 024CD 111         A=R1                   A(A)= chip addr.
    129 024D0 131         D1=A                   Chip addr to D1.
    130 024D3 3400        LCHEX  04000           Size of chips in bytes.
            040
    131 024DA CA          A=A+C  A               New chip address
    132 024DC CA          A=A+C  A                 back to R1.
    133 024DE 101         R1=A
    134 024E1 CE          C=C-1  A               One less byte for counter.
    135 024E3 D1          B=0    A               B(B)= checksum.
    136             *
    137             *---------------- Checksum loop --------------------------
    138 024E5 14B  CKSUM3 A=DAT1 B           :   Read byte...
    139 024E8 A68         B=A+B  B           :     add to checksum.
    140 024EB 540         GONC   CKSUM5      :
    141 024EE E5          B=B+1  A           :   If carry, increment checksum.
    142 024F0 171  CKSUM5 D1=D1+ 2           :   To next byte.
    143 024F3 CE          C=C-1  A           :   Count bytes.
    144 024F5 5FE         GONC   CKSUM3      :
    145             *-------------------------------------------------------
    146             *
    147 024F8 3174        LCASC  \G\             "G" for GOOD.
    148 024FC DA          A=C    A               A(B)= ASCII "G".
    149 024FE 118         C=R0                   Flag bits saved in R0(2).
    150 02501 AC6         C=A    S               Copy chip # to C(S) for beeps.
    151 02504 3133        LCHEX  33              Control for high freq.
    152 02508 CD          B=B-1  A
    153 0250A 969         ?B=0   B               Checksum should=0.
    154 0250D 90          GOYES  ROMCK3          GOOD chip.
    155 0250F 315E        LCHEX  E5              Control for low freq.
    156 02513 B0A         A=A-C  P               A(B)= ASCII "B".
    157 02516 C6   ROMCK3 C=C+C  A               Freq, set flag if necessary.
    158 02518 108         R0=C                   Save freq, chip# for beeps.
    159 0251B 7950        GOSUB  DspBld          Display "G" or "B", build dsp.
    160 0251F 3102        LCASC  \ \             Display trailing blank.
    161 02523 DA          A=C    A               A(B)= ASCII blank.
    162 02525 7F40        GOSUB  DspBld
```

```
163                *                                Medium beep, followed by
164                *                                  1,2,3 or 4 result beeps.
165                *
166 02529 3178             LCHEX  87                Control for medium beep.
167 0252D 7910  ROMCK5     GOSUB  RCKBp
168 02531 B37   ROMCK7     D=D+1  X                 Delay between beeps.
169 02534 5CF              GONC   ROMCK7
170                *
171 02537 118              C=R0                     Counter to C(S).
172 0253A A4E              C=C-1  S                 More beeps?
173 0253D 108              R0=C                     (Replace counter.)
174 02540 5CE              GONC   ROMCK5            More beeps.
175                *
176 02543 111              A=R1                     A(S)= chip counter.
177 02546 605F             GOTO   CHKSUM            Next chip.
178                *
179                *
```

```
                        EJECT
                *****************************************************************
                *****************************************************************
                **
                ** Name:    RCKBp   -  ROM Check Beep (Chirp utility)
                **
                ** Category:   GENUTL
                **
                ** Purpose:
                **      Non-clocked beep utility; no Flag(-2) check.
                **
                ** Entry:
                **      C(1)= frequency control (see below)
                **      C(0)= duration control (see below)
                **
                ** Exit:
                **      Carry clear, C(X)=00F, beeper off.
                **      P      =  0
                **
                ** Uses:
                **      A(A), B(A), C(A), D(W), P, OUT register.
                **
                ** Stk lvls:   0
                **
                ** Detail:
                **      Let  f= control nibble passed in C(1).
                **           d= control nibble passed in C(0).
                **      and  F= 60+33*f
                **
                **      Then   Frequency= (CPU KHZ/2)/F
                **      and    Duration = F*(256-16*d)/(CPU KHZ)
                **
                **        note: CPU KHZ is usually 650000 to 700000.
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------   ---------------------------------
                **  05/13/83   MB           documentation
                **
                *****************************************************************
                *****************************************************************
0254A D5        =RCKBp  B=C    A              Freq control to B(0).
0254C F1                BSL    A              B(XS)= freq, B(B)= duration.
0254E D4                A=B    A              Save freq in A(XS).
02550 D2                C=0    A
02552 28                P=     8              To toggle beeper.
                *
                *                       CPU cycles:
                *                       -----------
02554 801        RCKBp5 OUT=C                   6   Toggle beeper
02557 80F2              CPEX   2               6
0255B BF3        RCKBp7 DSL    W               19
0255E A2D               B=B-1  XS              4   Frequency control.
02561 59F               GONC   RCKBp7          10
```

```
235                *
236 02564 D8               B=A     A           7   Restore freq.
237 02566 B64              A=A+1   B           5   Duration.
238 02569 5AE              GONC    RCKBp5      10
239 0256C 20     =EXITBP   P=      0
240 0256E 32F0             LCHEX   00F             Leave beep low.
          0
241 02573 801              OUT=C
242 02576 03               RTNCC
243                *
```

```
244                     EJECT
245                *********************************************
246                *
247 02578 8E00  DspBld GOSUBL =DSPCHA        Display char.
          00
248 0257E 8C00         GOLONG =BLDDSP       Build display.
          00
249                *
250                *************************************************************
251                *
252 02584 B1    =RmTsMs NIBHEX B1              "Escape E, Escape <".
253 02586 54            NIBASC \E\
254 02588 B1            NIBHEX B1              "ROM TEST".
255 0258A C325          NIBASC \<ROM TES\
          F4D4
          0245
          5435
256 0259A 4502          NIBASC \T \
257 0259E FF            NIBHEX FF
258                *
259 025A0               END
```

BF2BLD	Ext		-	95				
BLDDSP	Ext		-	248				
CHKSM1	Abs	9417	#024C9 -	127	109			
CHKSUM	Abs	9367	#02497 -	102	177			
CKSUM3	Abs	9445	#024E5 -	138	144			
CKSUM5	Abs	9456	#024F0 -	142	140			
DSPCHA	Ext		-	247				
DspBld	Abs	9592	#02578 -	247	127	159	162	
=EXITBP	Abs	9580	#0256C -	239				
=RCKBp	Abs	9546	#0254A -	222	167			
RCKBp5	Abs	9556	#02554 -	230	238			
RCKBp7	Abs	9563	#0255B -	232	234			
ROMC19	Abs	9403	#024BB -	119	114			
ROMC21	Abs	9409	#024C1 -	122	123			
ROMCHK	Abs	9348	#02484 -	93				
ROMCK3	Abs	9494	#02516 -	157	154			
ROMCK5	Abs	9517	#0252D -	167	174			
ROMCK7	Abs	9521	#02531 -	168	169			
=ROMTST	Abs	9348	#02484 -	92				
=RmTsMs	Abs	9604	#02584 -	252	94			

Input Parameters

  Source file name is MB&ROM::MS

  Listing file name is MB/ROM:TI:ML::-1

  Object file name is MB%ROM:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *        J  PPPP    &     PPPP   RRRR     1
 2            *        J  P   P  & &    P   P  R   R   11
 3            *        J  P   P  & &    P   P  R   R    1
 4            *        J  PPPP    &     PPPP   RRRR     1
 5            *        J  P      & & &  P      R R      1
 6            *    J   J  P      &  &   P      R  R     1
 7            *     JJJ   P       && &  P      R   R   111
 8            *
 9                      TITLE  Line Parse Driver - Part 1 <831212.1516>
10 025A0               ABS    #025A0
11                     RDSYMB TIXEQU::MS
12                     RDSYMB SBXRAM::MS
13            *
14            ****************************************************************
15            ****************************************************************
16            **
17            ** Name:(S) LINEP   -  Parse Main Driver after ENDLINE
18            ** Name:(S) LINEP+  -  Parse Main Driver from anywhere
19            ** Name:(S) LNPEXT  -  Parse Main Driver external entry
20            ** Name:(S) LNEP66  -  Parse Main Driver return entry
21            **
22            ** Category:   PARUTL
23            **
24            ** Purpose:  Main driver routine to parse a line:
25            **           1) LINEP entry is called by MAINLP after
26            **              ENDLINE is entered on an input line.
27            **           2) LINEP+ entry is called to parse a
28            **              line, regardless of where the line is.
29            **              Used by direct execute keys (colon
30            **              key definitions) and STARTUP.
31            **           3) LNPEXT entry is the 'external parse'
32            **              entry.  By setting flRTN, it ensures
33            **              that in all cases (including errors),
34            **              control returns to the caller.  Used
35            **              by TRANSFORM.
36            **
37            ** Entry:    3 entry points:
38            **         1) LINEP  - Line to be parsed is in the display
39            **                     buffer.
40            **         2) LINEP+ - INBS points to start of input line.
41            **         3) LNPEXT - External Parse Entry
42            **                     Needed statuses (including S13) should
43            **                     be saved.  INBS points to start of
44            **                     input line.  OUTBS points to where
45            **                     tokenized line should go.
46            **                     AUTINC should be zero - may be default
47            **
48            ** Exit:
49            **       LINEP:
50            **          If valid program statement(s)
51            **             It is edited into current program file
52            **          If valid calculator BASIC statement(s)
53            **            (including implied DISP)
54            **             It is executed
55            **          Else ERROR exit
```

```
                **              Error message displayed;
                **              Line redisplayed with cursor;
                **              Jump to MAINLP
                **        LNPEXT:
                **            S5=1 => Line# on line
                **            S5=0 => No line#
                **            Carry clear => Line parsed successfully. Compiled
                **                           line starts at address pointed to
                **                           by OUTBS.
                **                           Compiled line length in R3.
                **            Carry set   => Error in parse.
                **                           C(3-0) = error#.
                **                           If C(3-0) = 0000
                **                             Then found only tEOL ("null line")
                **                            (May be preceded by a line#; S5
                **                            indicates presence of a line#)
                **
                **        NOTE: Any usage of LNPEXT entry rules out implied
                **              DISP in the case of failed implied LET parse.
                **
                ** Calls:     GNXTCR, LIN#P, NTOKEN, NTOKNL, CRGJMP, I/OAL+,
                **           OUT2TK, RANGE, EXPPAR, EXPEXC, MAKEBF, RTNSET,
                **           FILEP+, PEDIT, MOVEUA, SYCOLL, WSR0-3, AVS=D0
                **           CRLFOF,OVFLCK, TRNFCK, D1=IBS, OBCOLL, LDCSET,
                **           AUTCLR, LBLCK, PEDITD, SURSTU, RESPTR, OUTB+5,
                **           FSPC12, !CK, !CK3, RS-R03, OUT3TK, OUT1TK, WRDSCN
                **           STMTL+, UPDIN+, OUTBYT, ELSEP, LNP00, OBLCMP,GETLEE
                **
                ** Uses:      A-D, R0-R3, D1,D0, S0-S11,
                **           S-R0-2, S-R0-3, STMTR1 (all 16 nibbles), STMTD0
                **           flRTN (only used with LNPEXT entry)
                **
                **  Stk Lvls: 7
                **
                ** NOTES: A) Line parse only special checks for TRANSFORM
                **           (external entry) in four distinct places:
                **           1) eol,
                **           2) line#, followed by eol
                **           3) parse error
                **           4) correctly parsed line about to be edited into
                **              program memory.
                **
                **        B) Implied DISP isn't legal immediately after THEN/ELSE
                **
                **        C) Any usage of LNPEXT entry rules out implied DISP in
                **           the case of failed implied LET parse.  For example:
                **           10 5*A    would be parsed as:
                **           10 DISP 5*A
                **           But:
                **           10 A*5    would result in an error.
                **
                ** Detail:
                **      Key RAM and CPU register usage:
                **          S-R1-0 Original error# before 1st RESTART
                **          S-R1-1 Original Error position before 1st RESTART
```

```
111          **         S-R0-2 (Subr Save)                                GLOBAL
112          **         S-R0-3 IF clause in progress                      GLOBAL
113          **         S-R1-2 (RESTART ADDR), S-R1-3 (RESTART FLAG)      GLOBAL
114          **         STMTDO (RESTART PTR)                              GLOBAL
115          **             S4 - No restore of input pointer              GLOBAL
116          **             S5 - Line number found, program stmt          GLOBAL
117          **             S6 - Pending THEN                             GLOBAL
118          **             S7 - Multi-statement line                     TEMP
119          **                  Always CLEARED by EXPPAR call            GLOBAL
120          **             S8 - Delete (for PEDIT)                       TEMP
121          **             S9 - Middle of IF  (for ERROR)                TEMP
122          **            S10 - Implied LET Error                        GLOBAL
123          **             R3 - Error Msg Ptr & Line position if IMPLET Err
124          **              D - End of available memory
125          **
126          **       Available status for a Parse routine:  S8, S9
127          **       These 2 status bits are clear on entry for all
128          **       parse routines.
129          **
130          **  Algorithm:
131          **
132          **     Entry point for TRANSFORM (LNPEXT) saves return
133          **     stack level in S-R0-2 and sets flRTN  => A:
134          **
135          **  LINEP: (normal statement parse entry point)
136          **     Copy Display Buffer to Command Stack (MAKEBF)
137          **     Set INBS to start of input line in command stack
138          **     Send Carriage Return & Line Feed  (CRLFOF)
139          **      (so next character will clear display buffer)
140          **     Clear externally invoked flag (flRTN)
141          **
142          ** A:  Set OUTBS to AVMEMS (Collapses Output buffer)
143          **     Point D1 to start of input line
144          **     Clear S0-S11, S13
145          **     Set D(A) = End of Available Memory
146          **          D0 = OUTBS (Output buffer start)
147          **     Call Block 1
148          **
149          **     Retokenize lexeme
150          **     If line#
151          **       Set S5; Decrement D0 (delete statement
152          **       length byte at buffer start); Output line#
153          **       Call Block 5
154          **       If tEOL
155          **         If externally invoked (flRTN set)
156          **           THEN error
157          **           ELSE clear AUTO flag; delete line
158          **  B:   Decrement D0
159          **       Call 1.
160          **       Retokenize.
161          **
162          ** B1:   If Begin BASIC command (S3=1)
163          **         THEN goto E.
164          **         ELSE If System Command (S3=0,S0=1)
165          **                THEN error
```

```
166             ** C:          If !
167             **               THEN parse remark; goto 12
168             **               ELSE error.
169             **      If externally invoked (flRTN set)
170             **        THEN error;
171             **      Clear AUTO flag
172             **      If tEOL (null line)
173             **        THEN exit parse
174             **        ELSE goto C.
175             **
176             **      BLOCK 1:
177             **      Save D0 (statement length byte) in INADDR;
178             **      Increment D0; Clear RESTART flag (S-R1-3);
179             **      Clear Err# (S-R1-0); Call NTOKEN;
180             **      Set RESTART flag if XWORD or XFN &
181             **      save RESTART address (S-R1-2).
182             **      Save contents of LEXPTR (position of D1
183             **      before NTOKEN call) in STMTD0 - will be
184             **      needed to restore input pointer for RESTART.
185             **      Clear Middle of IF flag (S9) - Allows Implied
186             **       LET error to recover as Implied DISP
187             **
188             ** Entry point for variable or FN after THEN/ELSE:
189             **
190             ** C2:  If variable or FN:
191             **        set implied LET error flag (S10)
192             **        If no line# on line
193             **          Clear AUTO flag
194             **  G:    Try implied LET parse
195             **        Goto 10.
196             **      If looking at 1st lexeme on line
197             **        If line# followed by !
198             **          set S5; output line#; save D0 (location of
199             **          statement length byte) in INADDR; increment
200             **          D0; Parse remark; goto 12
201             **      If not a terminator (eg not tEOL,@,!,tELSE)
202             **        If legal implied DISP statement followed by
203             **        a terminator
204             **          If no line number on line
205             **               Clear AUTO flag; goto 10:
206             **      Restore D1,D0; return
207             **      END OF BLOCK 1
208             **
209             **  ***Block 5 only returns if a label is not found***
210             **
211             **      BLOCK 5
212             **      Save D0 (position of statement length byte) in
213             **      INADDR; increment D0
214             **      If quote
215             **        Set appropriate flag(s);
216             **        Step over it; Call FILEP+
217             **        If legal
218             **          THEN If matching closing quote
219             **   8:             THEN if colon follows
220             **                       THEN LEGAL LABEL;
```

```
221          **                              Output tLBLST & label
222          **                              If tEOL follows
223          **                                THEN goto 13
224          **                                ELSE goto 11 (parse as @)
225          **                          ELSE RESPTR; Return
226          **                    ELSE RESPTR; Return
227          **              ELSE RESPTR; Return
228          **          If 1st character is letter
229          **            RESPTR; GNXTCR; FILEP1; Goto 8
230          **          END BLOCK 5
231          **
232          **  D:    If not Calculator BASIC (S0=0)
233          **          THEN If begin BASIC (S3=1)
234          **                 THEN error
235          **                 ELSE goto C.
236          **  E:    If in IF statement (S-R0-3 nonzero)
237          **  F:      If not legal after THEN/ELSE (S2=0)
238          **            THEN error
239          **          If pending THEN (S6=1)
240          **            If token is IF token
241          **              THEN error
242          **
243          **          If XWORD
244          **            THEN Output 3-byte token
245          **            ELSE Output 1-byte token
246          **          Calculate Parse address
247          **          Clear flags (S0,S8,S9,S10)
248          **          Gosub to Parse routine   (CRGJMP)
249          **          If Middle of IF return  (Carry Set)
250          **            THEN Extended IF token already output;
251          **                 INADDR points to following byte;
252          **                 D0 is pointing past that byte
253          **                 S9 is set (middle of IF flag)
254          **                 S-R0-3 nonzero (IF in progress)
255          **   H:            If S5=1
256          **                   THEN goto B1
257          **                   ELSE goto D
258          **
259          **  10: Normal stmt return (carry clr)
260          **          Get Next Token
261          **          If ELSE
262          **            If no pending THEN (S6=0)
263          **              THEN error
264          **              ELSE Clear S6; Decr D0; Output t@;
265          **                   Call STMTLN, UPDIN+; Output tELSE
266          **                   Call ELSEP; goto 10
267          **          Check legal stmt terminators (@,!,EOL)
268          **           Clear S7
269          **           If @  (Multi-statement line)
270          **  11:        THEN Set S7, Output t@
271          **              ELSE If !   (Remark)
272          **                     THEN Output t!, Remark; goto 12
273          **                     ELSE If EOL
274          **  12:                      THEN Output tEOL
275          **                           ELSE Error Exit --> Excessive Chars
```

```
276           ** 13:  Output terminator
277           **       Clear S10 (Implied LET error flag)
278           **       Calculate & write out statement length
279           **       If multi-statement line
280           **         If S5=1
281           **           THEN Call 5; Goto B
282           **           ELSE Call 1; Goto D
283           **       Set AVMEMS to D0
284           **       If line# found (S5=1)
285           **         If externally invoked (flRTN set)
286           **           THEN exit with carry clear
287           **           ELSE Edit line into program memory (PEDIT)
288           **                Return to Main Loop
289           **       Calculate output buffer length, move to I/O buffer
290           **       area; call SYCOLL (Resets AVMEMS,OUTBS to SYSEN)
291           **       Execute Calc. BASIC Stmt (BSCEXC)
292           **
293           **      See the portion of the algorithm handled in IFP
294           **      in JP&PR3
295           **
296           ** History:
297           **
298           **    Date     Programmer   Modifications
299           **  --------   ----------   -------------
300           **  07/08/82   S.W.         Updated documentation
301           **  10/15/82   S.W.         Added call to D1=IBS
302           **  01/07/83   S.W.         Added algorithm
303           **  06/03/83   JP           Set AVMEMS @ D0 before PEDITD call
304           **  11/01/83   S.W.         Modified documentation header.
305           **
306           ****************************************************************
307           ****************************************************************
308           *
309           ****************************************************************
310           ****************************************************************
311           **
312           ** Name:(S) pPARSE  -  Parse Take Over Poll
313           **
314           ** Category:   POLL
315           **
316           ** Type:       FPOLL
317           **
318           ** Purpose:
319           **      Parse take-over to allow a LEX file to parse an input
320           **       line as other than BASIC
321           **
322           ** Should poll be "Handled"
323           **      Don't worry about XM, since if handled, there's no return
324           **
325           ** Meaning of "Handling" Poll (what does code do if handled?):
326           **      Parses line, acts accordingly, returns to MAINLP.
327           **
328           ** Entry conditions for handler (registers, ST, RAM, etc.):
329           **      Carry set
330           **      B[A] = Poll number.
```

```
331             **      HEX mode.
332             **      P=0.
333             **      INBS points to input line
334             **
335             ** Normal exit conditions from handler if handled (ST, RAM,
336             ** registers, etc.):
337             **      Return to MAINLP
338             **
339             ** Normal exit conditions from handler if not handled (ST, RAM,
340             ** registers, etc.):
341             **      HEX mode.
342             **      XM=1.
343             **
344             ** Available subroutine levels:
345             **      5
346             **
347             ** NOTE:
348             **
349             **      --SCRATCH RAM TO CONSIDER BELOW:--
350             **      --STMT/FN Scratch, SCRTCH, SNAPBF, TRFMBF, LDCSPC,--
351             **      --LEXPTR.--
352             **
353             ** What registers/RAM may be used if handled?:
354             **      A-D, D0, D1, P
355             **      R0-R4, S0-S11, STMT/FN scratch
356             **
357             ** What registers/RAM may be used if not handled?:
358             **      A-C, D[15-5] D0, D1, P
359             **      R0-R4, S0-S11, STMT/FN scratch
360             **
361             ** Special memory/pointer considerations (are pointers funny?):
362             **      No
363             **
364             ** Envisioned application(s):
365             **      'Auto Comment'
366             **      Alternate language parse (in conjunction with pEDIT)
367             **
368             ** History:
369             **
370             **    Date     Programmer             Modification
371             **  --------   ----------   ----------------------------------
372             **  02/15/83   S.W.         Added poll
373             **
374             ****************************************************************
375             ****************************************************************
376             *
377             ****************************************************************
378             ****************************************************************
379             **
380             ** Name:    SVRST   -  Save Lex Analysis Restart Information
381             **
382             ** Category:   PARUTL
383             **
384             ** Purpose:
385             **      1) Updates INADDR, increments D0
```

```
386             **      2) Clears RESTART flag (S-R1-3), Clears original err#
387             **         (S-R1-0), Calls NTOKNL
388             **      3) Sets RESTART flag if XWORD, XFN, or FFN
389             **      4) Saves RESTART addr in S-R1-2; saves LEXPTR
390             **         contents in STMTDO and saves D0 in R3
391             **
392             ** Entry:
393             **      4 entry points:
394             **      1) SVRSTU - Does 1-4 above. D1 at optional leading
395             **         blanks prior to lexeme.
396             **      2) SVRST+ - Does 2-4 above. Entry same as SVRSTU
397             **      3) SVRST2 - Does 3-4 above. Requires exit conditions
398             **                  from RESCAN:
399             **                  C(A)   = RESTART address
400             **                  A      = Tokenized lexeme
401             **                  LEXPTR   contains address prior to last
402             **                           lexeme scanned.
403             **      4) SAVRS+ - Does 4 above. C(A) and LEXPTR same as
404             **         required for SVRST2 above.
405             **
406             ** Exit:
407             **      D0 preserved from entry
408             **      STMTDO = LEXPTR contents after last NTOKEN call.
409             **      A(S) contains value of RESTART flag (S-R1-3)
410             **
411             ** Calls:      UPDINA, R.STPR, NTOKNL, XTOKCK
412             **
413             ** Uses.......
414             **  Exclusive: A,C,S-R1-2, S-R1-3
415             **  Inclusive: A,C,S-R1-2, S-R1-3, STMTDO
416             **
417             ** Stk lvls:   3
418             **
419             ** Detail:
420             **      Register A cannot be used here (it contains the TOKEN)
421             **
422             ** History:
423             **
424             **    Date     Programmer             Modification
425             **  --------   ----------   ----------------------------------
426             **  07/08/82   JP           Modified documentation
427             **  09/02/82   S.W.         Tweaked 2 lower entry points
428             **
429             ***************************************************************
430             ***************************************************************
431             *
432 025A0 8E00  SVRSTU GOSUBL =UPDINA
          00
433             *
434 025A6 AC0  =SVRST+ A=0    S
435 025A9 7024        GOSUB  R.STPR         Clear RESTART Flag
436 025AD D0          A=0    A
437 025AF 1F18        D1=(5) =S-R1-0        Err# (RESTARTed yet?)
          8F2
438 025B6 141         DAT1=A A
```

```
439 025B9 135          D1=C                 Restore D1 (Saved in R.STPR)
440 025BC 850          ST=1   0
441 025BF 8E00         GOSUBL =NTOKNL
          00
442 025C5 06   =SVRST2 RSTK=C               Save RESTART address
443 025C7 75F3         GOSUB  XTOKCK        Set RESTART flag if needed
444 025CB 07           C=RSTK               Restore RESTART address
445               *
446 025CD 136   SAVRS+ CDOEX
447 025D0 06           RSTK=C               SAVE D0
448 025D2 136          CDOEX                RESTORE C(A)
449 025D5 1BB8         D0=(5) =S-R1-2
          8F2
450 025DC 144          DAT0=C A             WRITE OUT RESTART ADDR
451 025DF 07           C=RSTK
452 025E1 134          D0=C                 RESTORE D0
453               *
454 025E4 137          CD1EX                Preserve D1
455 025E7 06           RSTK=C
456 025E9 1FFC         D1=(5) =LEXPTR
          6F2
457 025F0 147          C=DAT1 A             Input position after blanks skipped
458 025F3 1E19         D1=(4) =STMTD0
          8F
459 025F9 145          DAT1=C A
460 025FC 07           C=RSTK               Restore D1
461 025FE 135          D1=C
462 02601 01           RTN
463               *
464 02603 1BB7 =RTNSET D0=(5) =S-R0-2
          8F2
465 0260A 144          DAT0=C A
466 0260D 314D         LC(2)  =flRTN
467 02611 8C00         GOLONG =sflags       Set flRTN flag
          00
468               *
469               * ENTRY POINT FOR TRANSFORM (& others)
470               *
471 02617 07   =LNPEXT C=RSTK               SAVE RTN ADDR IN SCRATCH RAM
472 02619 76EF         GOSUB  RTNSET          & Set flRTN
473 0261D 5E1          GONC   LNPTR2        (B.E.T.)
474               *
475               * NORMAL ENTRY POINT FOR LINE PARSE
476               *    MAKEBF copies display to command stack
477               *
478 02620 8E00 =LINEP  GOSUBL =MAKEBF        Copy display to command stack
          00
479               *
480               * SECONDARY ENTRY POINT
481               *     Assumes C @ start of buffer to Parse
482               *
483               * Set INBS = C
484               * Send CR/LF
485               *
486 02626 1B6C =LINEP+ D0=(5) =INBS          Input Buffer Start
```

```
          6F2
487 0262D 144          DATO=C A              Point INBS at input buffer
488 02630 7000         GOSUB  =CRLFOF        Send CR/LF
489               *
490 02634 8E00         GOSUBL =FPoll
          00
491 0263A 4F           CON(2) =pPARSE        Parse take-over
492               *
493               * Collapse Output Buffer
494               * Set D1 = Input Buffer start
495               * Clear Status
496               * Set D  = Available memory end
497               * Set D0 = Output Buffer start
498               *
499 0263C AC2   LNPTR2 C=0    S              Clear global IF flag
500 0263F 8E00         GOSUBL =WSR0-3        S-R0-3
          00
501 02645 8E00         GOSUBL =OBCOLL        Collapse Output Buffer
          00
502 0264B 8E00         GOSUBL =D1=IBS        Set D1 = (INBS)
          00
503 02651 08           CLRST                 Clear S0-S11
504 02653 84D          ST=0   13             Pgm running (in case of error)
505 02656 8E00         GOSUBL =LDCSET        Set up D(A) & D0
          00
506               *
507 0265C 7292         GOSUB  LNP05          Line# allowed
508 02660 850          ST=1   0
509 02663 7973         GOSUB  LIN#P
510 02667 502          GONC   LNEP00         Line# found?
511               *
512               * No line#
513               *    If null line
514               *       If externally invoked
515               *          rtn with C(3-0)=0000
516               *       Goto MAINLP
517               *    Continue parse
518               *
519 0266A 7933         GOSUB  AUTCLR         No line#=>clr AUTO flag
520 0266E 3100         LC(2)  =tEOL          Load EOL token
521 02672 966          ?A#C   B              Not Null line ?
522 02675 F0           GOYES  lnep26
523 02677 7965         GOSUB  TRNFCK
524 0267B 454          GOC    LNEP01         Not called by TRANSFORM?
525 0267E 06    TRANSF RSTK=C
526 02680 D2           C=0    A              Err# = 0000
527 02682 02           RTNSC                 Carry set=> error
528               *
529 02684 6DE0  lnep26 GOTO   LNEP26         Continue
530               *
531               * Line # found - Set Line# found flag
532               *                Output Line#
533               *                Save Input buffer position for RESTART
534               *                Save Restart address for RESTART
535               *                If next token = EOL
```

```
                *                  Delete line / End AUTO mode
                *               else
                *                  If next token = "
                *                     Set quote flag
                *                     Parse as label
                *                  Save line length position
                *
02688 855   LNEP00 ST=1   5              Set line # found flag
0268B 181          D0=D0- 2
0268E 7000         GOSUB  =OUT2TK       Output line number
02692 7F20         GOSUB  LBLCK
02696 14B          A=DAT1 B
02699 31D0         LCHEX  0D            Load EOL character
0269D 962          ?A=C   B             EOL ?
026A0 60           GOYES  LNPEOL        DELETE line and clr AUTO mode
026A2 67B0         GOTO   LNEP07
                *
                * Line# followed by EOL
                *      If externally invoked
                *         exit with C(3-0)=0000 and carry set
                *         Line# found flag is set (S5)
                *      Set AVMEMS @ D0 so PEDIT works
                *      Clear AUTO flag & DELETE line
                *
026A6 7A35  LNPEOL GOSUB  TRNFCK
026AA 53D          GONC   TRANSF
                *
026AD 8E00         GOSUBL =AVS=D0        Set AVMEMS @ D0
      00
026B3 70F2         GOSUB  AUTCLR        Clr AUTO flag
026B7 858          ST=1   8              Set Delete Flag
026BA 8F00         GOSBVL =PEDITD       Line number with null line
      000
026C1 62D1  LNEP01 GOTO   LNEP99        Return to Main Loop
                *
026C5 77DE  LBLCK  GOSUB  SVRSTU
026C9 848          ST=0   8              CLEAR 'GENERAL' QUOTE FLAG
026CC 849          ST=0   9              CLEAR SINGLE QUOTE FLAG
026CF 3122         LCASC  \"\
026D3 962          ?A=C   B
026D6 81           GOYES  LBLCK"
026D8 300          LC(1)  =a'
026DB 962          ?A=C   B
026DE D0           GOYES  LBLCK'
026E0 7093         GOSUB  resptr         Position at 1st character
026E4 5C0          GONC   LBLCK1         (B.E.T.)
                *
026E7 6C83  LBLRST GOTO   resptr         Back up ptr to start of
                *                        supposed label
026EB 859   LBLCK' ST=1   9              SINGLE QUOTE FLAG
026EE 858   LBLCK" ST=1   8              GENERAL QUOTE FLAG
                * Quote found - Check as label
026F1 8E00  LBLCK1 GOSUBL =FILEP+        Parse label
      00
026F7 5FE          GONC   LBLRST         Not an initial letter?
```

```
588 026FA AF8          B=A    W              Save label name
589             *
590             * Save address of stmt length byte
591             * Skip over length byte
592             * If no line number (Calc. Basic), skip over label check
593             *
594             * If possible quoted label (S8=1)
595             *    Label Parse is already done
596             *    Test return status for FILEP-
597             *
598             *
599             * If variable
600             *    Restore pointer
601             *    Position to first non-blank character
602             *    Parse label
603             *          If no closing quote, error exit
604             *          Output label token, skip label chain field
605             *          Output label name
606             *          If next char = EOL
607             *             then goto Output EOL, Parse next statement
608             *             else goto Output @, Treat as Multi-statement line
609             *       If illegal label
610             *          Restore pointer
611             *
612             * NOTE: LABELED STATEMENTS ARE NOW ALLOWED IN MULTI-STATEMENT
613             *       IF/THEN/ELSE COMMANDS
614             *
615 026FD 14B          A=DAT1 B               Check closing quote
616 02700 868          ?ST=0  8               No beginning quote?
617 02703 91           GOYES  LNPLB+
618 02705 3122         LCASC  \"\
619 02709 869          ?ST=0  9               Double quote?
620 0270C 50           GOYES  CL"CK
621 0270E 307          LC(1)  \'\
622 02711 966   CL"CK  ?A#C   B
623 02714 24           GOYES  LNEPO6
624 02716 171          D1=D1+ 2               Step over closing quote
625 02719 14B          A=DAT1 B
626             *
627 0271C 31A3  LNPLB+ LCASC  \:\
628 02720 966          ?A#C   B               Not colon
629 02723 33           GOYES  LNEPO6
630 02725 3100  LNPLBL LC(2)  =tLBLST         Label Statement token
631 02729 8E00         GOSUBL =OUTB+5         Output label token & 5 nib field
          00
632 0272F 171          D1=D1+ 2               Step over colon
633 02732 AF4          A=B    W               Move filename to C
634 02735 8E00         GOSUBL =FSPC12         Output label name
          00
635 0273B 8E00         GOSUBL =GNXTCR
          00
636 02741 31D0         LCHEX  0D              Carriage Return ?
637 02745 962          ?A=C   B               End of statement ?
638 02748 60           GOYES  LNP5.7
639 0274A 65E0         GOTO   LNP721          Treat as multi-stmt
```

```
640 0274E 3100  LNP5.7 LC(2)  =tEOL          EOL Token
641            *        A=C    B
642 02752 6401         GOTO   LNEP76         Output, Parse next line
643            *
644            * No label on front of line
645            *    Restore data pointer to before label parse
646            *    Save Input buffer position for RESTART  (R3)
647            *
648 02756 7A13  LNEP06 GOSUB  resptr          Restore input pointer
649 0275A 181   LNEP07 D0=D0- 2               Compensate for UPDINA
650            *                              done on failed LABELP
651            * Get next token
652            *
653 0275D 7E81         GOSUB  LNP00
654            * Can replace next instruction w/ ST=1 0 & NTOKEN call
655            * More code, but more time efficient
656 02761 7F54         GOSUB  ntken+
657            *
658            * LINE# FOUND - DON'T ALLOW 'SYSTEM COMMAND'
659            * ASSUME THAT A SYSTEM COMMAND IS:
660            *                                   NOT BEGIN BASIC,
661            *                                   BUT IS CALC BASIC
662 02765 873  =LNEPLN ?ST=1  3
663 02768 32           GOYES  LNEP50          BEGIN BASIC?
664 0276A 860          ?ST=0  0               NOT SYSTEM COMMAND?
665 0276D F0           GOYES  LNEP30
666            *
667 0276F 592          GONC   LNEPE6          (B.E.T.) Sys Comm on numbered line
668            *
669            *
670            * No Line Number
671            *    Check if Calculator BASIC token
672            *
673 02772 870  =LNEP26 ?ST=1  0               Calculator BASIC token ?
674 02775 61           GOYES  LNEP50
675 02777 873          ?ST=1  3               Begin BASIC=> Illegal context
676 0277A F1           GOYES  LNEPE6            Trap out DATA, IMAGE, etc
677            *
678 0277C 7000  LNEP30 GOSUB  =!CK            Allow comment on front
679 02780 460          GOC    LNEP27          No comment?
680 02783 6660         GOTO   LNEP66
681            * S10 set IFF immediately after THEN/ELSE
682 02787 6000  LNEP27 GOTO   =ERRDSP         Want coherent cursor position
683            *                              of unnumbered line
684            * Line # found - Legal Begin BASIC
685            *
686            * If after THEN/ELSE, Check legality
687            * If XWORD keyword
688            *    Output all 3 XWORD tokens
689            * else
690            *    Output BASIC token
691            * GOSUB to Parse Address
692            *    Use Execution Address in B to find Parse Address
693            *    4 nibbles above Execution Address is Parse Address for keywo
694            *    Set High Nib of Parse Address from D0(4)
```

```
695             *
696             * RESTART re-entry
697             *    RESTAR has been jumped to by XWORD
698             *    NTOKEN has been done; Token = Begin BASIC | Calc BASIC | Sys
699             *
700 0278B 7484  LNEP50 GOSUB  RS-R03         Test S-R0-3
701 0278F 94A          ?C=0   S
702 02792 91           GOYES  LNEP55        Not in an IF clause?
703             *
704             * If after THEN/ELSE; Check if Legal
705             *
706 02794 872   LNEP52 ?ST=1  2             LEGAL AFTER THEN/ELSE?
707 02797 60           GOYES  LNEP54
708 02799 65B3 LNEPE6 GOTO   ONPE3          ERR06 - ILLEGAL AFTER THEN/ELSE
709             *
710             *
711 0279D 866   LNEP54 ?ST=0  6             NO PENDING THEN?
712 027A0 B0           GOYES  LNEP55
713             *
714 027A2 3100         LC(2)  =tIF
715 027A6 962          ?A=C   B
716 027A9 0F           GOYES  LNEPE6        NO IF AFTER THEN (ONLY AFTER ELSE)
717             *
718 027AB 3100  LNEP55 LC(2)  =tXWORD       Check if XWORD token
719 027AF 966          ?A#C   B             Not XWORD ?
720 027B2 90           GOYES  LNEP57
721 027B4 7000         GOSUB  =OUT3TK       Output XWORD,ROM ID,Entry#
722 027B8 560          GONC   LNEP58        B.E.T.
723             *
724 027BB 7000  LNEP57 GOSUB  =OUT1TK       Output BASIC keyword
725             *
726             * Calculate Parse Address
727             *
728 027BF D9    LNEP58 C=B    A             C=Exec Addr
729 027C1 136          CDOEX                DO=Exec Addr, C=Old DO
730 027C4 184          DO=DO- 5             5 nibbles up is PARSE address
731 027C7 142          A=DATO A             Read rel PARSE address
732 027CA 136          CDOEX                C=Parse Address, DO restored
733 027CD C2           C=C+A  A             Absolute Parse Address
734             *
735             * Gosub to Parse Routine
736             *
737 027CF 840          ST=0   0             Clear Characterization flag
738 027D2 848          ST=0   8             Clear for use by parse routines
739 027D5 849          ST=0   9             " "
740 027D8 84A          ST=0   10            Clear Implied LET error
741                                          since jumping on legal token
742 027DB 8E00         GOSUBL =CRGJMP       Jump to individual parse routine
          00
743 027E1 580   LNEP60 GONC   LNEP66        Normal statement return
744             *
745             * Middle of IF statment
746             *
747 027E4 8C00  LNEP61 GOLONG =LINE#?        Try comment before giving up
          00
```

```
748                *
749                * All other statements return from PARSE
750                *
751 027EA 7000 =LNEP66 GOSUB  =WRDSCN        Get EOL
752 027EE 00           CON(2) =tELSE
753 027F0 800          REL(3) LNEP67
754 027F3 00           CON(2) 0
755                *
756 027F5 5E2          GONC   LNEP68         tELSE NOT FOUND (B.E.T.)
757                *
758                * tELSE FOUND
759                * If pending THEN  (S6)
760                *    Save address of statement length byte
761                *       Calculate & write out length of THEN
762                *       Output ELSE token
763                *       Gosub to ELSE Parse
764                *       If extended ELSE
765                *          Go process next statement
766                *       else
767                *          error
768                *
769 027F8 866   LNEP67 ?ST=0  6              NO PENDING 'THEN'?
770 027FB E9           GOYES  LNEPE6         => Illegal context error
771 027FD 846          ST=0   6              CLEAR PENDING 'THEN' FLAG
772 02800 181          D0=D0- 2
773 02803 3100         LC(2)  =t@
774                * Clearing S10 fixes bug: IF <expr> THEN A=B ELSE KEY 1
775 02807 7000         GOSUB  =STMTL+        Outbyt byte & calc. stmt length
776 0280B 7000         GOSUB  =UPDIN+        UPDATE INADDR
777                *
778 0280F 3100         LC(2)  =tELSE
779 02813 7000         GOSUB  =OUTBYT
780                *
781 02817 8E00         GOSUBL =ELSEP
            00
782 0281D 46C          GOC    LNEP61         CARRY=>NOT LINE#,LBL, OR ASSIGNMENT
783 02820 73A3         GOSUB  Ntoken
784                *
785                * Check for legal statement terminators
786                *
787 02824 847   LNEP68 ST=0   7
788 02827 3104         LCASC  \@\
789 0282B 966          ?A#C   B
790 0282E C0           GOYES  LNEP73
791                *
792                * If Multi-Statement Line
793                *    Set Status  (S7)
794                *
795 02830 3100  LNP721 LC(2)  =t@             THEN TOKEN
796 02834 857          ST=1   7              MULTI-STMT LINE
797 02837 5F1          GONC   LNEP76         (B.E.T.)
798 0283A 3112  LNEP73 LCASC  \!\
799 0283E 966          ?A#C   B
800 02841 60           GOYES  LNEP74
801                *
```

```
802             * Parse REMARK at end of line
803             *
804 02843 7000          GOSUB  =!CK3          TERMINATES ON 0D ONLY
805 02847 3100  LNEP74  LC(2)  =tEOL
806 0284B 5B0           GONC   LNEP76         RTN FROM REMP=>CARRY CLR
807             *
808             * If not EOL after successful Parse
809             *    Excessive Characters Error
810             *
811 0284E 962           ?A=C   B                  EOL ?
812 02851 60            GOYES  LNEP75
813 02853 6000          GOTO   =ERR08         ERROR - EXCESSIVE CHARACTERS
814             *
815             * Legal statement termination
816             *
817             * Calculate statement length
818             * If multi-statement line --> Go back and parse next stmt
819             * If calculator statement --> Execute it
820             *
821             * If at tEOL, S13 will automatically be cleared;
822             *   either before call to BSCEXC (S5=0) or by MAINLP (S5=1)
823             *
824 02857       LNEP75
825             * Clear S10 (Implied LET error), Output byte, Calc stmt length
826 02857 7000  LNEP76  GOSUB  =STMTL+
827             *
828 0285B 867           ?ST=0  7              NOT A MULTI-STATEMENT LINE?
829 0285E B1            GOYES  LNEP85
830             *
831             * MULTI-STATEMENT LINE - VERIFY NEXT TOKEN IS LEGAL BEGIN BASIC
832             *
833             * If Label Start  (" | ')
834             *    go Parse Label
835             * else
836             *    go Parse as Statement
837             *
838 02860 875           ?ST=1  5              LINE#?
839 02863 E0            GOYES  LNEP78
840 02865 7680          GOSUB  LNP00
841 02869 7753          GOSUB  ntken+
842 0286D 640F          GOTO   LNEP26
843 02871 705E  LNEP78  GOSUB  LBLCK
844 02875 64EE          GOTO   LNEP07
845             *
846             * Legal end of statement found
847             *   Set AVMEMS @ Current D0 = (End of Output Buffer)
848             *
849 02879 8E00  LNEP85  GOSUBL =AVS=D0        Set AVMEMS @ D0
          00
850             *
851             * Parsed okay
852             * If externally invoked, then exit with carry clear
853             *
854 0287F 7163          GOSUB  TRNFCK
855 02883 06            RSTK=C
```

```
    856 02885 500          RTNNC                  CARRY CLR=> externally invoked
    857               *
    858 02888 865          ?ST=0  5               Execute Calculator BASIC?
    859 0288B F0           GOYES  LNEP87
    860 0288D 8F00         GOSBVL =PEDIT          Edit new line into program
              000
    861               *
    862               * MAINLP with reset AVMEMS and OUTBS to SYSEN
    863               *
    864 02894 8C00  LNEP99 GOLONG =MAINLP         Return to Main Loop
              00
    865               *
    866               * Calculator BASIC Statement
    867               *    Compute length of Output Buffer
    868               *    Move statement (Output Buffer) to Statement Buffer
    869               *      Expand Statement Buffer with NO LEEWAY check
    870               *      On exit from I/OAL+:
    871               *         D1   @ Start of Statement Buffer
    872               *         A=D1 on exit
    873               *         B(A) = Length of Output Buffer
    874               *    Reset Available Memory Start, Output Buffer Start
    875               *
    876 0289A 8E00  LNEP87 GOSUBL =OBLCMP         Compute length Output Buffer
              00
    877 028A0 D8           B=A    A               Length of Output Buffer
    878 028A2 3210         LC(3)  =bSTMT          Statement Buffer ID
              8
    879 028A7 8F00         GOSBVL =I/OAL+         Expand Stmt Buffer w/o LEEWAY check
              000
    880 028AE 460          GOC    LNEP88
    881 028B1 6000         GOTO   =OUTOVF         GOTO MEMERR: Insufficient Memory
    882               *
    883               * Entry to MOVEUA
    884               *   D1     @ Start of destination  bSTMT  (from I/OAL+)
    885               *   A(A)   @ Start of source       OUTBS
    886               *   AVMEMS = End of source
    887               *
    888 028B5 137   LNEP88 CD1EX                  Statement Buffer Start
    889 028B8 06           RSTK=C                 SAVE PTR TO STMT start
    890 028BA 137          CD1EX                  Restore D1
    891 028BD 1BF8         D0=(5) =OUTBS          Output Buffer start
              5F2
    892 028C4 142          A=DAT0 A               Start of source
    893 028C7 8F00         GOSBVL =MOVEUA         MOVE OUTPUT BUFFER TO I/O AREA
              000
    894 028CE 8E00         GOSUBL =SYCOLL         Reset AVMEMS, OUTBS to SYSEN
              00
    895               *
    896               *    Jump to BSCEXC to Execute statement line
    897               *         D0 = Start of Statement line
    898               *         BSCEXC will return to MAINLP when done executing
    899               *
    900               * Collapse command stack if it's into LEEWAY
    901 028D4 1B08         D0=(5) =RAWBFR
              5F2
```

```
902 028DB 146          C=DAT0 A
903 028DE 8E00         GOSUBL =GETLEE        Get LEEWAY back
          00
904             *
905 028E4 07           C=RSTK                Saved start of stmt buffer
906 028E6 134          D0=C                  @ Statement Length of line to execu
907             * S13 cleared at start of parse
908 028E9 8C00         GOLONG =BSCEXC        Execute Calc BASIC line
          00
909             *
910             ********************************************************************
911             ********************************************************************
912             **
913             ** Name:    LNP00   -  Line Parse Driver Utility
914             **
915             ** Category:   PARUTL
916             **
917             ** Purpose:
918             **      Traps out various possibilities for 1st token on a line
919             **       or first token after a line number, an '@', or a label:
920             **      Handles:
921             **         Implied LET of variables and user-defined functions.
922             **         Line number followed immediately by !.
923             **         Implied DISP (only if it's not a 'null' statement)
924             **
925             ** Entry:
926             **      P      =  0
927             **      D1 at start of line, or after label, @, or line#
928             **      D0 past last token written to output buffer
929             **      D(A) = (AVMEME)
930             **       3 entry points:
931             **          1) LNP00 - used if no line# is allowed (used by all
932             **                     callers, except for 1st token on line)
933             **          2) LNP05 - Used if 1st token on line. S8=0
934             **          3) LNP06 - Used immediately after THEN/ELSE to
935             **                     parse implied LET.
936             **                     Entry conditions as per NTOKEN exit:
937             **                     Requires that last NTOKEN returned a
938             **                     variable or tFN in register A.
939             **                     S9=1 => Don't try implied DISP parse if
940             **                             parse errors.
941             **
942             ** Exit:
943             **      No return to caller if any one of the following:
944             **           1) Implied LET of a variable or FN
945             **           2) Line number followed by '!'
946             **           3) Implied DISP
947             **         (Parse concludes on these statements, and control
948             **          is handed directly over to the parse driver).
949             **
950             **       else returns to caller (LNP00 and LNP05 entry only):
951             **           D1,D0 restored to what they were on entry
952             **           S-R1-2 set with RESTART address of 1st token found
953             **           S-R1-3 (RESTART flag) set iff token was XWORD or
954             **                  XFN
```

```
 955            **          R3=D0; (STMTD0) = D1
 956            **
 957            ** Calls:      SVRSTU, FNP+j, IMLETP, OUT2TK, LIN#P, UPDINA, !CK,
 958            **             EOLCK+, OUTBYT, RESPTR, CKTRN+, CKTRNS, NTOKEN,
 959            **             DSPP02
 960            **
 961            ** Uses.......
 962            **  Exclusive: A-C,R0,R1, S-R1-2,S-R1-3,STMTD0,S8,S9
 963            **  Inclusive: A-C,R0,R1,R3, S-R1-2,S-R1-3,STMTD0,S0-S3,S8,S9,S11
 964            **
 965            ** Stk lvls:   6
 966            **
 967            ** Note:       Falls into AUTCLR code.  This routine must
 968            **             immediately precede AUTCLR.
 969            **
 970            ** Algorithm:
 971            **      Traps out cases in following order:
 972            **       (NR => No return to caller - stmt parse completed)
 973            **       1) Implied LET of variables & FN (NR)
 974            **       2) Line number followed by '!'   (NR)
 975            **       3) Implied DISP                  (NR)
 976            **       4) If none of 1-3 above, LNP00 returns with D1,D0
 977            **          restored to what they were on entry.
 978            **           (includes case of a 'null' statement: tEOL, t@,
 979            **            t!, or tELSE found - LNP00 doesn't assume
 980            **            implied DISP for an empty statement)
 981            **
 982            ** History:
 983            **
 984            **    Date     Programmer             Modification
 985            **  --------   ----------   ---------------------------------
 986            **  09/02/82   S.W.         Wrote routine
 987            **
 988            ************************************************************************
 989            ************************************************************************
 990            *
 991 028EF 858   LNP00  ST=1   8              No line# allowed
 992 028F2 7AAC  LNP05  GOSUB  SVRSTU
 993 028F6 849          ST=0   9              Try Implied DISP if error
 994 028F9 87B  =LNP06  ?ST=1  11             Variable?
 995 028FC 71           GOYES  LNP10
 996 028FE 3100         LC(2)  =tFN
 997 02902 966          ?A#C   B
 998 02905 91           GOYES  LNP50
 999 02907 7490         GOSUB  CKTRN+         Set S10 (Implied LET)
1000 0290B 7ED1         GOSUB  FNP+j
1001 0290F 6ADE  LNEP36 GOTO   LNEP66
1002 02913 7880  LNP10  GOSUB  CKTRN+         Set S10 & clr AUTO if no line#
1003 02917 76D1         GOSUB  IMLETP
1004 0291B 53F          GONC   LNEP36         (B.E.T.)
1005           *
1006           * NOT IMPLIED LET - TRY IMPLIED DISP AFTER TRAPPING
1007           * OUT: Line numbers followed by !; Terminators
1008           *
1009 0291E 136   LNP50  CDOEX                 Save D0
```

```
1010 02921 134          D0=C
1011 02924 10B          R3=C
1012 02927 878          ?ST=1  8             Line# not allowed?
1013 0292A E2           GOYES  LNP52
1014 0292C 76B0         GOSUB  LIN#P2
1015 02930 472          GOC    LNP52         Not a line#?
1016 02933 181          D0=D0- 2
1017 02936 7000         GOSUB  =OUT2TK       Output line#
1018            * TRAP OUT    10 !
1019 0293A 7982         GOSUB  Ntoken
1020 0293E 3112         LCASC  \!\
1021 02942 966          ?A#C   B             NOT comment?
1022 02945 01           GOYES  LNP51
1023 02947 855          ST=1   5
1024 0294A 7000         GOSUB  =UPDINA
1025 0294E 7000         GOSUB  =!CK
1026 02952 5CB          GONC   LNEP36        LINE# FOLLOWED BY COMMENT (B.E.T.)
1027            *
1028 02955 181   LNP51  D0=D0- 2
1029 02958 7E11  LNP52  GOSUB  EOLCKR        Restore ptr & look for end stmt
1030 0295C 4D2          GOC    LNP60         End stmt token?
1031            *
1032 0295F 7D20         GOSUB  LNP63         Restore D1
1033 02963 858          ST=1   8
1034 02966 3100         LC(2)  =tDISP
1035 0296A 7000         GOSUB  =OUTBYT
1036 0296E 8E00         GOSUBL =DSPP02
          00
1037 02974 451          GOC    LNP60         ERROR ON IMPLIED DISP?
1038 02977 7301         GOSUB  =EOLCK+
1039 0297B 5E0          GONC   LNP60
1040            * LEGAL IMPLIED DISP - NO LINE#
1041 0297E 7020         GOSUB  CKTRNS        Clr AUTO
1042 02982 7EE0 =LNP55  GOSUB  resptr        Restore pointer (D1)
1043 02986 636E         GOTO   LNEP66
1044            * RESTORE D1,D0 - D0 saved above
1045            *                 D1 saved during SAVRST call
1046 0298A 11B   LNP60  C=R3
1047 0298D 134          D0=C
1048 02990 1F19  LNP63  D1=(5) =STMTD0
          8F2
1049 02997 147          C=DAT1 A
1050 0299A 135          D1=C
1051 0299D 01           RTN
1052            *
1053 0299F 85A   CKTRN+ ST=1   10            Implied LET flag
1054            * Implied DISP or possible Implied LET found
1055 029A2 875   CKTRNS ?ST=1  5             Line# already found?
1056 029A5 00           RTNYES
1057            *
1058            * No line#
1059            * Fall in AUTCLR
1060            *
1061            * NOTE: AUTCLR code must immediately follow
1062            *
```

```
1063                    STITLE AUTO Mode Clear
1064                ********************************************************************
1065                ********************************************************************
1066                **
1067                ** Name:    AUTCLR  -  Clear Auto Mode RAM Location
1068                **
1069                ** Category:   PARUTL
1070                **
1071                ** Purpose:
1072                **      Clears AUTINC RAM Location, indicating not in AUTO
1073                **      Mode.
1074                **
1075                ** Entry:  None
1076                **
1077                ** Exit:
1078                **          Carry Clear
1079                **          (AUTINC) = 0
1080                **
1081                ** Calls:      None
1082                **
1083                ** Uses.......
1084                **  Exclusive: C(A)
1085                **  Inclusive: C(A)
1086                **
1087                ** Stk lvls:   1
1088                **
1089                ** Note:
1090                **      This code fallen into from above
1091                **
1092                ** Detail:
1093                **      USES A SUBROUTINE LEVEL TO SAVE D1
1094                **
1095                ** History:
1096                **
1097                **    Date     Programmer              Modification
1098                **  --------   ----------   ----------------------------------
1099                **  07/07/82   JP           Modified documentation
1100                **  10/13/82   S.W.         Changed from W field to A field
1101                **
1102                ********************************************************************
1103                ********************************************************************
1104 029A7 137  =AUTCLR CD1EX
1105 029AA 06          RSTK=C
1106 029AC 1FBC        D1=(5) =AUTINC
           6F2
1107 029B3 D2          C=0    A
1108 029B5 15D3        DAT1=C 4
1109 029B9 07          C=RSTK
1110 029BB 135         D1=C
1111 029BE 03          RTNCC
1112                *
1113                ********************************************************************
1114                ********************************************************************
1115                **
1116                ** Name:    XTOKCK  -  Check for XWORD or XFN
```

```
1117                ** Name:    R.STPR  -  Sets or Clears RESTART Flag
1118                ** Name:    RSTPR1  -  Sets or Clears RESTART Flag
1119                **
1120                ** Category:   PARUTL
1121                **
1122                ** Purpose:
1123                **    XTOKCK entry checks token in A(B) and sets RESTART flag
1124                **           if it's an XWORD, XFN, or FFN token.
1125                **    R.STPR and RSTPR1 set or clear RESTART flag, depending
1126                **           on A(S).
1127                **
1128                ** Entry:
1129                **      3 entry points:
1130                **      1) XTOKCK - P=0. A(B) contains token.
1131                **      2) R.STPR
1132                **         RSTPR1 - A(S) = value to write to RESTART flag
1133                **
1134                ** Exit:
1135                **      P=0 (XTOKCK entry), else preserved from entry
1136                **      S-R1-3 nonzero iff XWORD, XFN, or FFN found (XTOKCK entry
1137                **      A(S) contains new value of S-R1-3 (RESTART flag)
1138                **      S-R1-3 contains A(S) on entry (R.STPR, RSTPR1 only)
1139                **
1140                ** Calls:      XCHECK
1141                **
1142                ** Uses.......
1143                **  Exclusive: A(S), C(A)
1144                **  Inclusive: A(S), C(A), S-R1-3,  D1 - RSTPR1 only
1145                **
1146                ** Stk lvls:   1
1147                **
1148                ** Note:       RESTART flag is RAM location S-R1-3
1149                **
1150                ** History:
1151                **
1152                **    Date     Programmer              Modification
1153                **  --------   ----------   ----------------------------------
1154                **  09/02/82   S.W.         Documented routine
1155                **  05/11/83   S.W.         Added call to XCHECK; now check for
1156                **                           funny functions
1157                **
1158                ********************************************************************
1159                ********************************************************************
1160 029C0 AC0    =XTOKCK A=0    S
1161 029C3 7000          GOSUB  =XCHECK
1162 029C7 500           RTNNC                  Not XWORD,XFN,or funny FN ?
1163 029CA A4C           A=A-1  S
1164                * SET RESTART FLAG
1165 029CD 137   =R.STPR CD1EX
1166 029D0 1F09  =RSTPR1 D1=(5) =S-R1-3
           8F2
1167 029D7 1514          DAT1=A S
1168 029DB 135           D1=C
1169 029DE 01            RTN
1170                *
```

```
1171            ********************************************************************
1172            ********************************************************************
1173            **
1174            ** Name:     LIN#P   -  Line Number Parse
1175            ** Name:     LIN#P2  -  Line Number Parse w/o NTOKEN call
1176            **
1177            ** Category:   PARUTL
1178            **
1179            ** Purpose:    Looks for tokenized line number
1180            **
1181            ** Entry:
1182            **             LIN#P  - D1 points at alleged Line#
1183            **             LIN#P2 - A(B) = alleged line# token
1184            **                      P=0
1185            **                      (Assumes NTOKNL has already been called)
1186            **
1187            ** Exit:       Carry Clear => Line number found
1188            **                            A(A) contains BCD Line# (NIB 4-0)
1189            **
1190            **             Carry Set   => Line number not found
1191            **
1192            ** Calls:      NTOKNL
1193            **
1194            ** Uses:       A, C(B) - LIN#P2 entry
1195            **             A-D, P, D1, R0, S0-S3, S11 - LIN#P entry
1196            **
1197            ** Stk Lvls:   0 - LIN#P2 entry
1198            **             3 - LIN#P  entry
1199            **
1200            ** History:
1201            **
1202            **    Date     Programmer   Modification
1203            **  --------   ----------   ------------
1204            **  11/01/83   S.W.         Cleaned up documentation header
1205            **
1206            ********************************************************************
1207            ********************************************************************
1208            *
1209            *
1210 029E0 8E00 =LIN#P  GOSUBL =NTOKNL
           00
1211 029E6 3100 =LIN#P2 LC(2)  =tLINE#
1212 029EA 966          ?A#C   B                NOT LINE # TOKEN?
1213 029ED 00           RTNYES
1214 029EF BF4          ASR    W
1215 029F2 F4           ASR    A                SHIFT OFF TOKEN
1216 029F4 03           RTNCC
1217            *
1218            *
```

```
1219                    STITLE GOTO/GOSUB Parse
1220                ********************************************************************
1221                ********************************************************************
1222                **
1223                ** Name:(S) GOTOp   -  GOTO Statement Parse
1224                ** Name:(S) GOSUBp  -  GOSUB Statement Parse
1225                **
1226                ** Category:   STPARS
1227                **
1228                ** Purpose:
1229                **      Parse GOTO | GOSUB statement
1230                **
1231                ** Entry:
1232                **      D1 past GOTO | GOSUB token
1233                **
1234                ** Exit:
1235                **      Carry Clear - If lineno | label is output
1236                **
1237                **       else error exit to PARERR:
1238                **                  Illegal first character: Syntax Error
1239                **
1240                ** Calls:      LBLINP
1241                **
1242                ** Uses.......
1243                **  Inclusive: A-C,D(S), S0-S3,S7,S9-S11, R0,R1,R3, P, D0,D1
1244                **
1245                ** Stk lvls:   6
1246                **
1247                ** Detail:
1248                **
1249                ** GOTOp:
1250                ** GOSUBp:  Parse lineno | label   (LBLINP)
1251                **          If carry set --> Error exit - "Syntax"
1252                **             else      --> RTNCC
1253                **
1254                ** History:
1255                **
1256                **    Date     Programmer              Modification
1257                **  --------   ----------   ---------------------------------
1258                **  07/08/82   JP           Modified documentation
1259                **
1260                ********************************************************************
1261                ********************************************************************
1262                *
1263 029F6          =GOTOp
1264 029F6          =GOSUBp
1265 029F6          =GOTOP
1266 029F6 7A00     =GOSUBP GOSUB  LBLINP         Parse lineno or label
1267 029FA 500              RTNNC                 Lineno or Label accepted
1268                *
1269 029FD 854      =LBLNPE ST=1   4              No restore of input pointer
1270 02A00 6000             GOTO   =ERR01         Error - Syntax
```

```
1271                    STITLE Label/Lineno Parse (LBLINP)
1272            ********************************************************************
1273            ********************************************************************
1274            **
1275            ** Name:(S) LBLINP  -  Parse Line Number or Label
1276            ** Name:(S) LBLNIF  -  Parse Line Number or Label after THEN/ELSE
1277            ** Name:(S) LINP    -  Parse Line Number only
1278            **
1279            ** Category:   PARUTL
1280            **
1281            ** Purpose:
1282            **      Parse line number or label:
1283            **      LBLINP or LBLNIF entry allows line number or label
1284            **      LINP entry looks for line number only
1285            **
1286            ** Entry:
1287            **       D0 points past last token written to output buffer
1288            **       D(A) contains (AVMEME)
1289            **       3 entry points:
1290            **       1) LBLINP - D1 pointing to alleged line# or label
1291            **       2) LINP   - D1 pointing to alleged line#. S9=1
1292            **       3) LBLNIF - Exit conditions from NTOKNL: P=0,
1293            **                   A(B) contains token to check, D1 past
1294            **                   alleged line# or label.
1295            **                   S9=0 => Allow line# or label
1296            **                   S9=1 => Allow line# only
1297            **
1298            ** Exit:
1299            **      Carry clear
1300            **            Line# or label found and tokenized
1301            **            D1 past line# or label
1302            **            D0 past tokenized line# or label
1303            **            If line# found,
1304            **               A(3-0) contains line#
1305            **               The following 11 nibbles are output:
1306            **               tLINE# 00000 <4 nib BCD line#>
1307            **            If label found, it is output in 1 of 2
1308            **               formats using either LABELP or FSPC10:
1309            **               tLBLRF <string expr>  - LABELP
1310            **               tLBLRF tLITRL <ascii label> - FSPC10
1311            **
1312            **      Carry set
1313            **            LBLINP entry => 1st char not letter | line#
1314            **            LINP   entry => Line# not found
1315            **            LBLNIF entry:
1316            **              S9=0 on entry => 1st char not letter | line#
1317            **              S9=1 on entry => Line# not found
1318            **
1319            ** Calls:      NTOKNL, LINP#2, LABELP, OUT3TK, OUT2TC
1320            **             OUTBYT, RESPTR, FSPC10 (golong)
1321            **
1322            ** Uses.......
1323            **  Exclusive: A,B,C,D1,S9,S10,D0,D1
1324            **  Inclusive: A,B,C,D1,S9,S10,D0,D1,S0-S3,S7,S11,D(S),R0,R1,R3,P
1325            **
```

```
1326            ** Stk lvls:   5
1327            **
1328            ** Detail:     S9 used by LBLINP entry only
1329            **            S10 used by LABELP to ensure no reserved word check
1330            **
1331            ** Algorithm:
1332            **
1333            **      If next token = line#  (LINP#2)
1334            **         Output line# token  (OUTBYT)
1335            **         Zero out Line# jump address field
1336            **         Output line#        (OUT1TK)
1337            **         Return, carry Clear
1338            **      If S9=1 (Line# Parse only)
1339            **         Return, carry Set
1340            **       else
1341            **         Output Label Reference Token  (OUTBYT)
1342            **         Restore Input pointer  (RESPTR)
1343            **         Set No RESERVE word parse flag (S10)
1344            **         Parse label  (LABELP)
1345            **         If legal label
1346            **            If string expression
1347            **               RTNCC  (Label already output)
1348            **            else
1349            **               golong to Output Literal Token & Label
1350            **         else
1351            **            Back up Output pointer over Label Token
1352            **            RTNSC   (Illegal first character found)
1353            **
1354            ** NOTE:
1355            **
1356            ** Tokenized form:
1357            **
1358            ** <lineno> ---> (Lineno Token) (5 nib jump addr) (4 nib Line#)
1359            ** <label>  ---> (Label Ref Token) (String Expression)
1360            ** <label>  ---> (Label Ref Token) (Literal Token) (ASCII Label)
1361            **
1362            ** History:
1363            **
1364            **    Date     Programmer             Modification
1365            **  --------   ----------   ---------------------------------
1366            **  07/08/82   JP           Modified documentation
1367            **  11/23/82   JP           Removed Stack level saving
1368            **  11/29/82   JP           Removed S2/Label found flag
1369            **  11/01/83   S.W.         Cleaned up documentation header
1370            **
1371            ********************************************************************
1372            ********************************************************************
1373 02A04 849  =LBLINP ST=0   9              WANT LABEL OR LINE#
1374            *
1375            * Check for line#
1376            *
1377 02A07 8E00 =LINP   GOSUBL =NTOKNL
           00
1378            *
1379            * IF Parse Entry
```

```
1380                *
1381                * If Line#   (LIN#P2)
1382                *    Output Line# token
1383                *    Zero out Line# address jump
1384                *      OUT3TK writes from A or will destroy A if using OUT3TC
1385                *      Therefore, must transfer Line# to C and use A for zeroing
1386                *    Output Line# from C
1387                *
1388 02A0D 75DF =LBLNIF GOSUB  LIN#P2          Line # Parse
1389 02A11 481         GOC    LBLN20          Not a Line#
1390 02A14 7000        GOSUB  =OUTBYT         Output Line# token
1391 02A18 AEE         ACEX   B               Restore A(B)
1392 02A1B D6          C=A    A               Move Line# to C
1393 02A1D D0          A=0    A               Zero field for line# addr jump
1394 02A1F 7000        GOSUB  =OUT3TK         Write 6 nibs from A to D0
1395 02A23 180         D0=D0- 1               Only want 5 nibs of zeroes
1396 02A26 6000        GOTO   =OUT2TC         Output line# from C, RTNCC
1397                *
1398                * If not Line#
1399                *    If Line# Parse request only
1400                *
1401 02A2A 879   LBLN20 ?ST=1  9               Line# parse only ?
1402 02A2D 00          RTNYES                 Return, carry set
1403                *
1404                * Check for Label
1405                *   Output Label Reference Token
1406                *     LABELP will output label if string expression
1407                *   Restore Input pointer
1408                *   Set No RESERVE word Check flag for Label Parse
1409                *
1410 02A2F 3100        LC(2)  =tLBLRF
1411 02A33 7000        GOSUB  =OUTBYT         Output Label Reference Token
1412 02A37 7000        GOSUB  =RESPTR         Restore Input pointer
1413 02A3B 85A         ST=1   10              No RESERVE word check
1414 02A3E 8E00        GOSUBL =LABELP         Parse label
           00
1415 02A44 516         GONC   EOLCK*          Label not found?
1416                *
1417                * If string variable
1418                *    RTNCC
1419                * else
1420                *    Output literal token and ASCII label
1421                *
1422 02A47 877         ?ST=1  7               STRING EXPRESSION?
1423 02A4A A2          GOYES  resptr
1424 02A4C 8C00        GOLONG =FSPC10         Output Literal token & label
           00
```

```
1425                    STITLE RESTORE Parse
1426            ********************************************************************
1427            ********************************************************************
1428            **
1429            ** Name:    RESTRP  -  RESTORE Statement Parse
1430            ** Name:(S) FIXP    -  FIX and WAIT Statement Parse
1431            **
1432            ** Category:   STPARS
1433            **
1434            ** Purpose:
1435            **      RESTRP parses RESTORE statement
1436            **
1437            **      FIXP parses FIX and WAIT statements.  It also parses a
1438            **      single numeric expression.
1439            **
1440            ** Entry:
1441            **      D(A) = (AVMEME)
1442            **      DO points into the output buffer
1443            **      RESTRP entry:
1444            **       D1 past RESTORE keyword
1445            **       DO past RESTORE token
1446            **      FIXP entry:
1447            **       D1 points at alleged numeric expression
1448            **
1449            ** Exit:
1450            **      RESTRP entry:
1451            **      Legal statement syntax =>
1452            **       Return with carry clear
1453            **       Statement parsed and tokenized
1454            **       D1 past legally parsed statement
1455            **       DO past token stream for RESTORE statement
1456            **       P=0
1457            **      Else take error exit
1458            **
1459            **      FIXP entry:
1460            **      Valid numeric expression found =>
1461            **       Return with carry clear
1462            **       Tokenized expression written to output buffer
1463            **       DO points past token stream
1464            **       D1 points immediately past the expression
1465            **      Else take error exit
1466            **
1467            ** Calls:      LBLINP,PILP+,WRDSCN, OUT1TK, RESPTR, NUMCK
1468            **             D1C=R3
1469            **
1470            ** Uses........
1471            **  Exclusive: R3,S8
1472            **  Inclusive: R3,S8,S0-S3,S7,SA11,A-C,D(S),D0,D1,R0-R3,
1473            **
1474            ** Stk lvls:   6
1475            **
1476            ** Detail:
1477            **      RESTORE [ <lineno> | <label> ]
1478            **      RESTORE [# <num expr> [, <num expr> ] ]
1479            **
```

```
1480                ** Algorithm:
1481                **
1482                **      Parse for lineno or label
1483                **      If lineno | label not found
1484                **         If channel # not found
1485                **            Return to main line parse to check for EOL
1486                **         else
1487                **            If comma follows Channel #
1488                **               Parse for <numeric expression>
1489                **      else
1490                **         RTNCC
1491                **
1492                ** History:
1493                **
1494                **    Date     Programmer              Modification
1495                ** --------   ----------   ----------------------------------
1496                **  07/08/82   JP           Modified documentation
1497                **  10/20/82   S.W.         No more RESTORE #<num expr>, END
1498                **
1499                ********************************************************************
1500                ********************************************************************
1501                *
1502                *
1503 02A52 7EAF   =RESTRP GOSUB  LBLINP         Parse lineno or label
1504 02A56 500          RTNNC                  Label | Line#
1505                *
1506                * IF NOT LINE# OR LABEL, THEN D1 WILL HAVE BEEN SAVED IN UPPER
1507                * PORTION OF R3 BY LABELP
1508                *
1509 02A59 7000         GOSUB  =D1C=R3         RESTORE INPUT PTR
1510 02A5D 858          ST=1   8               FLAG - EXIT AFTER COMMA
1511 02A60 8E00         GOSUBL =PILP+          LOOK FOR CHANNEL #
           00
1512 02A66 411          GOC    RSTRP7          NO '#'?
1513                *
1514                * SEE IF COMMA FOUND AFTER CHANNEL #
1515                *
1516 02A69 878          ?ST=1  8               NO COMMA AFTER CHANNEL #?
1517 02A6C 80           GOYES  resptr
1518                *
1519 02A6E              =WAITP
1520 02A6E 8E00         =FIXP  GOSUBL =NUMCK
           00
1521                *
1522 02A74 6000          resptr GOTO   =RESPTR
1523                *
1524 02A78 03            RSTRP7 RTNCC
1525                *
```

```
1526                    STITLE EOLCK - Check EOL,`,!,ELSE
1527                ********************************************************************
1528                ********************************************************************
1529                **
1530                ** Name:(S) EOLCK   -  Check for EOL,@,!,ELSE
1531                ** Name:(S) EOLCKR  -  Check for EOL,@,!,ELSE
1532                **
1533                ** Category:   PARUTL
1534                **
1535                ** Purpose:
1536                **      Checks for tEOL, @, ! , tELSE
1537                **      EOLCKR entry calls RESPTR before checking.
1538                **
1539                ** Entry:
1540                **      EOLCKR - NTOKEN (or WRDSCN) has already been
1541                **               called; D1 past keyword/character to
1542                **               check (except if token was tEOL)
1543                **      EOLCK  - D1 at optional blanks preceding
1544                **               keyword/character to check.
1545                **
1546                ** Exit:
1547                **      P=0
1548                **      A(B) = Token found
1549                **      D1 past the keyword/character found
1550                **      Carry Set =>
1551                **        Statement terminator found (tEOL, tELSE, @, !)
1552                **      Carry Clr =>
1553                **        Statement terminator not found
1554                **
1555                ** Calls:      WRDSCN, RESPTR
1556                **
1557                ** Uses.......
1558                **  Exclusive: A-C,D1,R1,R2,P
1559                **  Inclusive: A-C,D1,R0-R2,S0-S3,S11,P
1560                **
1561                ** Stk lvls:   4
1562                **
1563                ** Detail:     D0 is preserved from entry
1564                **
1565                ** History:
1566                **
1567                **    Date     Programmer             Modification
1568                **  --------   ----------   ---------------------------------
1569                **  07/08/82   JP           Modified documentation
1570                **  11/02/83   S.W.         Modified documentation - Routine
1571                **                           doesn't use D0.
1572                **
1573                ********************************************************************
1574                ********************************************************************
1575                *
1576                *
1577 02A7A 7000     =EOLCKR GOSUB  =RESPTR
1578                *
1579 02A7E          =EOLCK+
1580 02A7E 7000     =EOLCK  GOSUB  =WRDSCN
```

```
1581 02A82 00          CON(2) =tELSE
1582 02A84 220         REL(3) EOLCK*
1583 02A87 00          CON(2) 0
1584            *
1585 02A89 3100        LC(2)  =tEOL
1586 02A8D 962         ?A=C   B               EOL?
1587 02A90 00          RTNYES
1588 02A92 3104        LCASC  \@\             DON'T CK FOR KLUDGED TOKEN
1589 02A96 962         ?A=C   B
1590 02A99 00          RTNYES
1591 02A9B 3112        LCASC  \!\             "       "
1592 02A9F 962         ?A=C   B
1593 02AA2 00          RTNYES
1594 02AA4 01          RTN
1595            *
1596 02AA6 181   EOLCK* DO=DO- 2               BACK UP OVER tELSE
1597 02AA9 02          RTNSC
1598            *
```

```
1599                    STITLE BEEP Parse
1600            ********************************************************************
1601            ********************************************************************
1602            **
1603            ** Name:    BEEPP   -  BEEP Statement Parse
1604            ** Name:(S) DELAYp  -  DELAY and WINDOW Statement Parse
1605            **
1606            ** Category:   STPARS
1607            **
1608            ** Purpose:
1609            **      Parses BEEP, WINDOW and DELAY statements
1610            **
1611            ** Entry:
1612            **      D1 past BEEP, WINDOW, or DELAY keyword
1613            **      D0 past tBEEP, tWINDW, or tDELAY
1614            **      D(A) = (AVMEME)
1615            **
1616            ** Exit:
1617            **      Return with carry clear =>
1618            **       Accepted statement
1619            **
1620            **      Else error exit to PARERR
1621            **
1622            ** Calls:      NUMCK, COMCK1, RESPTR, OUT1TK, EOLCK
1623            **
1624            ** Uses.......
1625            **  Exclusive: A,C,D1,D0
1626            **  Inclusive: A-C,D(15-5),R0-R2,S0-S3,S7,S11,FUNCD0
1627            **             D1,D0
1628            **
1629            ** Stk lvls:   5
1630            **
1631            ** Detail:
1632            **      BEEP [ ON | OFF ]
1633            **      BEEP [ <frequency> [ , <duration> ]
1634            **
1635            **      DELAY <delayt> [,<scrollt>]
1636            **      WINDOW <start> [,<end>]
1637            **
1638            **      frequency, duration, delayt, scrollt, start, and
1639            **      end are all specified using numeric expressions.
1640            **
1641            ** Algorithm:
1642            **      If Next Token = End of Line Terminator
1643            **         Restore Pointer
1644            **         Return CC
1645            **      If Next Token = ON | OFF
1646            **         Output Token
1647            **         Return CC
1648            **      else
1649            **         Restore Input pointer
1650            **         Verify first parameter
1651            **         If next token = comma
1652            **            Verify second parameter
1653            **         else
```

```
1654                 **            Go Restore pointer & Return
1655                 **       RTNCC
1656                 **
1657                 ** History:
1658                 **
1659                 **    Date      Programmer            Modification
1660                 **  --------    ----------   ----------------------------------
1661                 **  07/08/82    JP           Modified documentation
1662                 **  08/18/82    S.W.         Combined WINDOW and DELAY parse with
1663                 **                            BEEP parse
1664                 **  11/01/83    S.W.         Modified documentation header.
1665                 **
1666                 ********************************************************************
1667                 ********************************************************************
1668                 *
1669 02AAB 7FCF  =BEEPP  GOSUB  EOLCK+
1670 02AAF 44C          GOC    resptr          Restore Ptr, Return if EOL
1671                 *
1672                 * Try BEEP ON | OFF
1673                 *
1674 02AB2 7000           GOSUB  =WRDSC+         Look for ON or OFF
1675 02AB6 00            CON(2) =tON
1676 02AB8 23D           REL(3) =LNEP66         Return CC
1677 02ABB 00            CON(2) =tOFF
1678 02ABD D2D           REL(3) =LNEP66         Return CC
1679 02AC0 00            NIBHEX 00
1680                 *
1681                 * Try <num exp> [ , <num exp> ]
1682                 *
1683 02AC2 7000           GOSUB  =RESPTR
1684 02AC6           =WINDWp
1685 02AC6 8E00  =DELAYp GOSUBL =NUMCK          Verify frequency
           00
1686 02ACC 8E00          GOSUBL =COMCK1         Check for comma
           00
1687 02AD2 51A           GONC   resptr          Not a comma
1688 02AD5 489           GOC    FIXP            (B.E.T.) Call NUMCK & RESTORE D1
1689                 *
```

```
1690                    STITLE LET/Implied LET Parse
1691            ********************************************************************
1692            ********************************************************************
1693            **
1694            ** Name:    LET     -  LET Statement Parse
1695            ** Name:    IMLETP  -  Implied LET Statement Parse
1696            **
1697            ** Category:   STPARS
1698            **
1699            ** Purpose:
1700            **      Parse LET and Implied LET statements.
1701            **      LETP entry is used when tLET is found.
1702            **      IMLETP entry is used when a variable or tFN is
1703            **       found at the beginning of a statement; if it
1704            **       doesn't turn out to be a syntactically correct
1705            **       implied LET, then depending on a status setting,
1706            **       we attempt to parse it as an implied DISP instead.
1707            **
1708            ** Entry:
1709            **      D(A) = (AVMEME)
1710            **      2 entry points:
1711            **      1) LETP   - D1 past LET keyword.
1712            **                  D0 past tLET.
1713            **      2) IMLETP - D1 past variable in input stream.
1714            **                  D0 immediately after stmt length byte.
1715            **                  S10=1
1716            **                  S9=1 => Don't parse as implied DISP if
1717            **                          an error occurs (Used by IF)
1718            **                  S9=0 => Parse as implied DISP if error
1719            **
1720            ** Exit:
1721            **      If accepted
1722            **         Return with carry clear
1723            **         P=0
1724            **         Statement tokenized and output
1725            **         D1 at end of legally parsed statement
1726            **         D0 past tokenized statement
1727            **         Implied LET Error flag is cleared (S10)
1728            **
1729            **      If unaccepted
1730            **         Control given to parse error handler (PARERR)
1731            **         If S10=1 and S9=0
1732            **            Then an implied DISP parse is attempted.
1733            **            Else error exit through MFERR.
1734            **
1735            ** Calls:      NTOKEN, EXPPLS, OUTBYT, RESPTR, GNXTCR, R3=D1*
1736            **             ASNP (goto)
1737            **
1738            ** Uses.......
1739            **  Exclusive: A-C, D0,D1, P
1740            **  Inclusive: A-C, R0-R3, D0,D1, S0-S3,S7-S11, P
1741            **
1742            **
1743            ** Stk lvls:   4
1744            **
```

```
1745            ** NOTE:
1746            **      IMLETP can't use S-R0-0.  This RAM location must be
1747            **      left intact from IF:  If implied LET parse fails
1748            **      after THEN/ELSE, then parse can recover with a
1749            **      label parse.
1750            **
1751            ** Detail:
1752            **      Status Bit Usage:
1753            **
1754            **      S7 = Left Side Evaluate for Expression Parse
1755            **      S8 = Numeric variable
1756            **      S9 = Middle of IF flag
1757            **      S9 = Don't try implied DISP parse if errors
1758            **      S10 = Implied LET Error
1759            **
1760            **      Syntax:
1761            **
1762            **      [ LET ] <[FN] numeric variable> = <numeric expression>
1763            **      [ LET ] <[FN] string  variable> = <string  expression>
1764            **
1765            **          (The = is tokenized with tCOMMA.)
1766            **
1767            ** Algorithm:
1768            **
1769            ** LETP: Get next token  (NTOKEN)
1770            **       If not variable (S11)
1771            **          If FN token
1772            **             Set S9 to indicate LET FN
1773            **                go Parse FN statement
1774            **          else
1775            **             Error ---> Illegal variable
1776            **IMLETP:
1777            **       Restore pointer to call Expr parse
1778            **       Set Left Hand side evaluation flag
1779            **       Call Expr Parse @ special entry pointer (EXPPLS)
1780            **       If dummy array       (Carry Set)
1781            **          Error ---> Illegal Variable
1782            **       If Value expression  (not Reference)
1783            **          Error ---> Illegal Variable
1784            **       Set Numeric Variable flag
1785            **       If illegal expression (S0)
1786            **          Error ---> Illegal Variable
1787            **       If string variable (S1)
1788            **          Clear numeric variable flag (S8)
1789            **       Restore input pointer
1790            **       Jump to ASNP
1791            **
1792            ** History:
1793            **
1794            **    Date     Programmer             Modification
1795            **  --------   ----------   ----------------------------------
1796            **  07/08/82   JP           Modified documentation
1797            **  08/31/82   S.W.         Commonized code with FNP
1798            **
1799            ************************************************************************
```

```
1800            ********************************************************************
1801            *
1802 02AD8 7BE0 =LETP   GOSUB  Ntoken
1803 02ADC 87B          ?ST=1  11              Variable?
1804 02ADF 21           GOYES  LETP10
1805            *
1806            * Check for LET FN
1807            *
1808 02AE1 3100         LC(2)  =tFN
1809 02AE5 966          ?A#C   B               not FN token ?
1810 02AE8 02           GOYES  LETPE4          Error - Illegal Variable
1811 02AEA 859          ST=1   9               No Implied DISP if error
1812 02AED 6000  FNP+j  GOTO   =FNP+           Parse FN statement
1813 02AF1      =IMLETP
1814 02AF1 8E00  LETP10 GOSUBL =R3=D1*         RESPTR,R3=D10
           00
1815 02AF7 857          ST=1   7               Set Left Side evaluate flag
1816 02AFA 8E00         GOSUBL =EXPPLS         Evaluate left side
           00
1817 02B00 470          GOC    LETPE4          Dummy array ?
1818 02B03 831          ?XM=0                  Not Value expr? (Reference?)
1819 02B06 60           GOYES  LETP15          O.K. on Left Hand Side
1820            *
1821 02B08 6000  LETPE4 GOTO   =ERR05          Illegal Variable
1822            *
1823 02B0C 858   LETP15 ST=1   8               Set numeric variable flag
1824 02B0F 873          ?ST=1  3               Valid numeric expression ?
1825 02B12 A0           GOYES  LETP20
1826 02B14 870          ?ST=1  0               Invalid expression ?
1827 02B17 1F           GOYES  LETPE4          Invalid variable Error
1828 02B19 848          ST=0   8               Clr Numeric Variable flag
1829            *
1830            * Look for "="
1831            * Set Implied LET Error flag; Clear if "=" found
1832            * Output Comma token in its place
1833            *
1834 02B1C 7000  LETP20 GOSUB  =RESPTR
1835 02B20 7000         GOSUB  =GNXTCR
1836            * Check for '=' & that right side of same type (string/numeric)
1837            * as left side - Assignment Parse
1838 02B24 6000         GOTO   =ASNMP
1839            *
```

```
1840                    STITLE ON Parse
1841            ********************************************************************
1842            ********************************************************************
1843            **
1844            ** Name:    ONP     -  ON Statement Parse
1845            ** Name:(S) ONP40   -  GOTO,GOSUB,RESTORE in middle of stmt Parse
1846            **
1847            ** Category:   STPARS
1848            **
1849            ** Purpose:
1850            **      Parse ON statement
1851            **
1852            **      Possible syntax is:
1853            **        ON ERROR (GOTO | GOSUB) (lineno | label)
1854            **        ON TIMER  # <timer no>,  <#secs> ( GOTO | GOSUB )
1855            **                  ( <lineno> | <label> )
1856            **        ON <exp> GOTO    <lineno>|<label> [, <lineno>|<label>]
1857            **             "   GOSUB          "
1858            **             "   RESTORE        "
1859            **
1860            ** Entry:
1861            **      D1 past ON keyword
1862            **      D0 past tON in output buffer
1863            **      D(A) = (AVMEME)
1864            **
1865            ** Exit:
1866            **      If accepted
1867            **        Return with carry clear
1868            **        P=0
1869            **        D1 past valid statement
1870            **        D0 past tokenized statement in output buffer
1871            **        S8=1 => ON ERROR | ON TIMER statement
1872            **
1873            **      If unaccepted
1874            **        Error exit through PARERR
1875            **
1876            ** Calls:       NUMCK,LBLINP,COMCK+,RESPTR,WRDSCN,NTOKEN
1877            **              #CK, NUMC++, RESPTR, CONCKO
1878            **
1879            ** Uses.......
1880            **  Exclusive: A,C,S8
1881            **  Inclusive: A,C,S8,B,D(15-5),S0-S3,S7,S11,R0-R3,FUNCD0
1882            **
1883            ** Stk lvls:   6
1884            **
1885            ** Detail:
1886            **
1887            **      ON <exp> ...        allowed from keyboard
1888            **      ON TIMER | ERROR    not allowed from keyboard
1889            **
1890            ** Algorithm:
1891            **
1892            **      If Next Token = ERROR
1893            **         If Keyboard execute --> Error exit
1894            **         Set ON ERROR statement flag
```

```
1895            **         goto 1;
1896            **      If Next Token = TIMER
1897            **         If Keyboard execute --> Error exit
1898            **         Set ON TIMER statement flag
1899            **         If next char # "#"
1900            **            Error Exit with No restore of input pointer
1901            **         Skip "#" and
1902            **         Verify <timer no> expression  (NUMC++)
1903            **         If A(B) # Comma  (F1)
1904            **            then Error ---> Syntax
1905            **         Output Comma token  (COMCKO)
1906            **         Verify <# secs> expression  (NUMCK)
1907            ** 1:   If Next Token # GOSUB | GOTO
1908            **         If ON-ERROR| TIMER stmt   ---> Error Exit
1909            **         If Token # RESTORE    ---> Error Exit
1910            ** 2:   Check for label | lineno  (LBLINP)
1911            **      If not label | lineno ---> Error Exit
1912            **      If ON-ERROR statement ---> RTNCC
1913            **      Check for comma and output (COMCKO)
1914            **        Continue Label/Lineno parse  (goto 2)
1915            **      else goto RESPTR (Position before non-comma & RTN)
1916            **
1917            ** History:
1918            **
1919            **    Date     Programmer             Modification
1920            **  --------   ----------   ----------------------------------
1921            **  07/08/82   JP           Modified documentation
1922            **  11/01/83   S.W.         Updated documentation header
1923            **
1924            ********************************************************************
1925            ********************************************************************
1926            *
1927 02B28 7000 =ONP    GOSUB  =WRDSCN
1928            *
1929            * Table for first keyword
1930            *
1931 02B2C 00           CON(2) =tERROR
1932 02B2E 110          REL(3) ONP10          ON ERROR
1933 02B31 00           CON(2) =tTIMER
1934 02B33 020          REL(3) ONP20          ON TIMER
1935 02B36 00           NIBHEX 00
1936            *
1937            * ERROR | TIMER Keyword not found
1938            * Try ON <exp>
1939            *
1940 02B38 7000         GOSUB  =RESPTR        Verify expression &
1941 02B3C 583          GONC   ONP35          try GOTO | GOSUB (B.E.T.)
1942            *
1943            * ON ERROR
1944            *
1945 02B3F 865   ONP10  ?ST=0  5              Keyboard execute ?
1946 02B42 D0           GOYES  ONPE3          ON ERROR not allowed
1947 02B44 858          ST=1   8              ON-ERROR flag
1948            *
1949            * Must get next token cus
```

```
1950            *       All other jumps to ONP40 have done NTOKEN
1951            *
1952 02B47 7C70          GOSUB   Ntoken          Get next token
1953 02B4B 6F20          GOTO    ONP40
1954 02B4F 6000  ONPE3   GOTO    =ERR06          Non-executable
1955            *
1956            * ON TIMER
1957            *
1958 02B53 865   ONP20   ?ST=0   5               Keyboard execute ?
1959 02B56 9F            GOYES   ONPE3           ON TIMER not allowed
1960 02B58 858           ST=1    8               Set ON TIMER flag
1961 02B5B 8E00          GOSUBL  =#CK            Is next char # ?
          00
1962 02B61 560           GONC    ONP30           Carry clear => Yes
1963 02B64 689E  lblnpe  GOTO    LBLNPE          No restore of input error
1964 02B68 8E00  ONP30   GOSUBL  =NUMC++         Skip "#"
          00
1965 02B6E 7000          GOSUB   =COMCK+
1966 02B72 5D1           GONC    ONPE1           Comma not found ?
1967            * Comma found & output
1968 02B75 8E00  ONP35   GOSUBL  =NUMCK          Verify <# secs>
          00
1969            *
1970            * Check for GOSUB | GOTO token
1971            * Call WRDSCN entry that does RESPTR before NTOKEN
1972            * NUMCK has done NTOKEN before RETURN
1973            *
1974 02B7B 7000 =ONP40   GOSUB   =WRDSC+
1975 02B7F 00            CON(2)  =tGOSUB
1976 02B81 810           REL(3)  ONP60           GOSUB
1977 02B84 00            CON(2)  =tGOTO
1978 02B86 310           REL(3)  ONP60           GOTO
1979 02B89 00            CON(2)  =tRESTR
1980 02B8B 900           REL(3)  ONP50
1981 02B8E 00            NIBHEX  00
1982            *
1983 02B90 6000  ONPE1   GOTO    =ERR01          Error --> Syntax
1984            *
1985 02B94 878   ONP50   ?ST=1   8               ON ERROR | TIMER ?
1986 02B97 9F            GOYES   ONPE1           Yes => Don't allow RESTORE
1987            *
1988            *    Check for lineno | label
1989            *
1990 02B99 776E  ONP60   GOSUB   LBLINP          Check for line# or label
1991 02B9D 46C           GOC     lblnpe          Set No Rstptr/Syntax error
1992 02BA0 878           ?ST=1   8               ON-ERROR ?
1993 02BA3 F1            GOYES   rtncc
1994            *
1995 02BA5 7000          GOSUB   =COMCK0         Check for comma & output
1996 02BA9 4FE           GOC     ONP60           Output comma and continue
1997 02BAC 6000          GOTO    =RESPTR         Restore to "non" comma, RTNCC
```

```
1998                    STITLE UNPROTECT Parse
1999                ********************************************************************
2000                ********************************************************************
2001                **
2002                ** Name:    UNPRTP  -  UNPROTECT Statement Parse
2003                **
2004                ** Purpose: Parse UNPROTEC(T)
2005                **
2006                ** Entry:   D1 past 'UNPROTEC'
2007                **          D0 past tUNPRT in output buffer
2008                **          D(A) = (AVMEME)
2009                **
2010                ** Exit:
2011                **          If valid syntax
2012                **           Return with carry clear
2013                **           D1 past legally parsed statement
2014                **           D0 past statement token stream
2015                **           P=0
2016                **          Else error exit through PARERR
2017                **
2018                ** Calls:    CNV2UC
2019                **
2020                ** Uses:     A,C, D0,D1, P
2021                **
2022                ** Stk Lvls: 2
2023                **
2024                ** History:
2025                **
2026                **    Date     Programmer    Modification
2027                **  --------   ----------   ---------------
2028                **  11/02/83   S.W.         New documentation header format adde
2029                **
2030                ********************************************************************
2031                ********************************************************************
2032                *
2033 02BB0 8E00    =UNPRTP GOSUBL =CNV2UC
           00
2034 02BB6 3145            LCASC  \T\
2035 02BBA 966             ?A#C   B
2036 02BBD 3D              GOYES  ONPE1
2037 02BBF 171             D1=D1+ 2              POINT PAST T
2038 02BC2 03     rtncc    RTNCC
2039                *
2040 02BC4 850     =ntken+ ST=1   0
2041 02BC7 8C00    =Ntoken GOLONG =NTOKEN
           00
2042                *
```

```
2043                    STITLE AUTO Mode Check
2044                ********************************************************************
2045                ********************************************************************
2046                **
2047                ** Name:    AUTOCK  -  AUTO Mode Check
2048                **
2049                ** Category:   PARUTL
2050                **
2051                ** Purpose:
2052                **      Indicates whether or not machine is in AUTO mode
2053                **
2054                ** Entry:
2055                **      None
2056                **
2057                ** Exit:
2058                **      Carry set => Not in AUTO mode
2059                **      Carry clr => In AUTO mode
2060                **                   C(A) = AUTO increment value - 1
2061                **
2062                ** Calls:      None
2063                **
2064                ** Uses.......
2065                **  Exclusive: A(A),C(A)
2066                **  Inclusive: A(A),C(A)
2067                **
2068                ** Stk lvls:   0
2069                **
2070                ** History:
2071                **
2072                **    Date      Programmer            Modification
2073                **  --------    ----------   ---------------------------------
2074                **  07/07/82    JP           Modified documentation
2075                **  10/13/82    S.W.         Changed from W field to A field
2076                **
2077                ********************************************************************
2078                ********************************************************************
2079 02BCD 133  =AUTOCK AD1EX
2080 02BD0 1FBC         D1=(5) =AUTINC
           6F2
2081 02BD7 D2           C=0    A
2082 02BD9 15F3         C=DAT1 4
2083 02BDD 131          D1=A
2084 02BE0 CE           C=C-1  A
2085 02BE2 01           RTN
2086                *
```

```
2087                    STITLE TRANSFORM Entry Check
2088                ********************************************************************
2089                ********************************************************************
2090                **
2091                ** Name:     TRNFCK  -  Check if line parse called by TRANSFORM
2092                **
2093                ** Category: PARUTL
2094                **
2095                ** Purpose:  Checks flRTN to see if line parse was called by
2096                **           TRANSFORM or by any other user of the special entry
2097                **           point, LNPEXT.
2098                **
2099                ** Entry:    P=0
2100                **
2101                ** Exit:     flRTN clear
2102                **           P=0
2103                **           CARRY SET => flRTN is clear
2104                **                        Entry was not through LNPEXT
2105                **                 CLR => flRTN was set
2106                **                        Entry was through LNPEXT
2107                **                        D1 destroyed from entry
2108                **                        C(A) address of caller
2109                **
2110                ** Calls:    SFLAGC
2111                **
2112                ** Uses:     C(S), C(A)
2113                **           If flRTN is set additionally uses:
2114                **             A(A), B(A), C(5), D(A), D1
2115                **
2116                ** Stack lvls: Carry set =>  0
2117                **                  clr =>  3
2118                **
2119                ** History:
2120                **
2121                **    Date     Programmer   Modifications
2122                **  --------   ----------   -------------
2123                **  02/04/83   S.W.         Updated documentation
2124                **
2125                ********************************************************************
2126                ********************************************************************
2127                *
2128                *
2129 02BE4 137  =TRNFCK CD1EX                  SAVE INPUT POINTER
2130 02BE7 1F3E        D1=(5) (=SYSFLG)+10    Pt to flRTN
           6F2
2131 02BEE 1574        C=DAT1 S
2132 02BF2 A46         C=C+C  S
2133 02BF5 581         GONC   NOTTRS
2134                * CARRY SET => CALLED BY TRANSFORM
2135                * Clear flRTN
2136 02BF8 314D        LC(2)  =flRTN
2137 02BFC 8E00        GOSUBL =sflagc
           00
2138 02C02 1FB7        D1=(5) =S-R0-2
           8F2
```

```
2139 02C09 147          C=DAT1 A              RETURN ADDRESS
2140 02C0C 03           RTNCC
2141              *
2142 02C0E 135   NOTTRS D1=C                  RESTORE INPUT POINTER
2143 02C11 02           RTNSC
2144              *
2145              ********************************************************************
2146              ********************************************************************
2147              **
2148              ** Name:    RS-R03  -  Read nibble in S-R0-3
2149              **
2150              ** Category:   GENUTL
2151              **
2152              ** Purpose:
2153              **      Reads contents of RAM location S-R0-3
2154              **
2155              ** Entry:
2156              **      None
2157              **
2158              ** Exit:
2159              **      C(S) contains contents of S-R0-3
2160              **
2161              ** Calls:      None.
2162              **
2163              ** Uses.......
2164              **      C(A), C(S)
2165              **
2166              ** Stk lvls:   0
2167              **
2168              ** History:
2169              **
2170              **    Date     Programmer              Modification
2171              **   --------   ----------   ----------------------------------
2172              **   02/07/83   S.W.         Added documentation
2173              **
2174              ********************************************************************
2175              ********************************************************************
2176              *
2177 02C13 137  =RS-R03 CD1EX                  Save D1
2178 02C16 1F08         D1=(5) =S-R0-3
           8F2
2179 02C1D 1574         C=DAT1 S
2180 02C21 135          D1=C                  Restore D1
2181 02C24 01           RTN
2182              *
```

```
 !CK     Ext                -   678  1025
 !CK3    Ext                -   804
 #CK     Ext                -  1961
 ASNMP   Ext                -  1838
=AUTCLR  Abs   10663 #029A7 -  1104   519   564
 AUTINC  Abs  194251 #2F6CB -    12  1106  2080
=AUTOCK  Abs   11213 #02BCD -  2079
 AVS=DO  Ext                -   563   849
=BEEPP   Abs   10923 #02AAB -  1669
 BSCEXC  Ext                -   908
 CKTRN+  Abs   10655 #0299F -  1053   999  1002
 CKTRNS  Abs   10658 #029A2 -  1055  1041
 CL"CK   Abs   10001 #02711 -   622   620
 CNV2UC  Ext                -  2033
 COMCK+  Ext                -  1965
 COMCK1  Ext                -  1686
 COMCK0  Ext                -  1995
 CRGJMP  Ext                -   742
 CRLFOF  Ext                -   488
 D1=IBS  Ext                -   502
 D1C=R3  Ext                -  1509
=DELAYp  Abs   10950 #02AC6 -  1685
 DSPP02  Ext                -  1036
 ELSEP   Ext                -   781
=EOLCK   Abs   10878 #02A7E -  1580
 EOLCK*  Abs   10918 #02AA6 -  1596  1415  1582
=EOLCK+  Abs   10878 #02A7E -  1579  1038  1669
=EOLCKR  Abs   10874 #02A7A -  1577  1029
 ERR01   Ext                -  1270  1983
 ERR05   Ext                -  1821
 ERR06   Ext                -  1954
 ERR08   Ext                -   813
 ERRDSP  Ext                -   682
 EXPPLS  Ext                -  1816
 FILEP+  Ext                -   586
=FIXP    Abs   10862 #02A6E -  1520  1688
 FNP+    Ext                -  1812
 FNP+j   Abs   10989 #02AED -  1812  1000
 FPoll   Ext                -   490
 FSPC10  Ext                -  1424
 FSPC12  Ext                -   634
 GETLEE  Ext                -   903
 GNXTCR  Ext                -   635  1835
=GOSUBP  Abs   10742 #029F6 -  1266
=GOSUBp  Abs   10742 #029F6 -  1264
=GOTOP   Abs   10742 #029F6 -  1265
=GOTOp   Abs   10742 #029F6 -  1263
 I/OAL+  Ext                -   879
=IMLETP  Abs   10993 #02AF1 -  1813  1003
 INBS    Abs  194246 #2F6C6 -    12   486
 LABELP  Ext                -  1414
 LBLCK   Abs    9925 #026C5 -   569   546   843
 LBLCK"  Abs    9966 #026EE -   584   574
 LBLCK'  Abs    9963 #026EB -   583   577
 LBLCK1  Abs    9969 #026F1 -   586   579
```

```
=LBLINP  Abs   10756 #02A04 -  1373  1266  1503  1990
 LBLN20  Abs   10794 #02A2A -  1401  1389
=LBLNIF  Abs   10765 #02A0D -  1388
=LBLNPE  Abs   10749 #029FD -  1269  1963
 LBLRST  Abs    9959 #026E7 -   581   587
 LDCSET  Ext                -   505
=LETP    Abs   10968 #02AD8 -  1802
 LETP10  Abs   10993 #02AF1 -  1814  1804
 LETP15  Abs   11020 #02B0C -  1823  1819
 LETP20  Abs   11036 #02B1C -  1834  1825
 LETPE4  Abs   11016 #02B08 -  1821  1810  1817  1827
 LEXPTR  Abs  194255 #2F6CF -    12   456
=LIN#P   Abs   10720 #029E0 -  1210   509
=LIN#P2  Abs   10726 #029E6 -  1211  1014  1388
 LINE#?  Ext                -   747
=LINEP   Abs    9760 #02620 -   478
=LINEP+  Abs    9766 #02626 -   486
=LINP    Abs   10759 #02A07 -  1377
 LNEP00  Abs    9864 #02688 -   543   510
 LNEP01  Abs    9921 #026C1 -   567   524
 LNEP06  Abs   10070 #02756 -   648   623   629
 LNEP07  Abs   10074 #0275A -   649   551   844
=LNEP26  Abs   10098 #02772 -   673   529   842
 LNEP27  Abs   10119 #02787 -   682   679
 LNEP30  Abs   10108 #0277C -   678   665
 LNEP36  Abs   10511 #0290F -  1001  1004  1026
 LNEP50  Abs   10123 #0278B -   700   663   674
 LNEP52  Abs   10132 #02794 -   706
 LNEP54  Abs   10141 #0279D -   711   707
 LNEP55  Abs   10155 #027AB -   718   702   712
 LNEP57  Abs   10171 #027BB -   724   720
 LNEP58  Abs   10175 #027BF -   728   722
 LNEP60  Abs   10209 #027E1 -   743
 LNEP61  Abs   10212 #027E4 -   747   782
=LNEP66  Abs   10218 #027EA -   751   680   743  1001  1043  1676  1678
 LNEP67  Abs   10232 #027F8 -   769   753
 LNEP68  Abs   10276 #02824 -   787   756
 LNEP73  Abs   10298 #0283A -   798   790
 LNEP74  Abs   10311 #02847 -   805   800
 LNEP75  Abs   10327 #02857 -   824   812
 LNEP76  Abs   10327 #02857 -   826   642   797   806
 LNEP78  Abs   10353 #02871 -   843   839
 LNEP85  Abs   10361 #02879 -   849   829
 LNEP87  Abs   10394 #0289A -   876   859
 LNEP88  Abs   10421 #028B5 -   888   880
 LNEP99  Abs   10388 #02894 -   864   567
 LNEPE6  Abs   10137 #02799 -   708   667   676   716   770
=LNEPLN  Abs   10085 #02765 -   662
 LNP00   Abs   10479 #028EF -   991   653   840
 LNP05   Abs   10482 #028F2 -   992   507
=LNP06   Abs   10489 #028F9 -   994
 LNP10   Abs   10515 #02913 -  1002   995
 LNP5.7  Abs   10062 #0274E -   640   638
 LNP50   Abs   10526 #0291E -  1009   998
 LNP51   Abs   10581 #02955 -  1028  1022
```

```
 LNP52    Abs   10584 #02958 -  1029  1013  1015
=LNP55    Abs   10626 #02982 -  1042
 LNP60    Abs   10634 #0298A -  1046  1030  1037  1039
 LNP63    Abs   10640 #02990 -  1048  1032
 LNP721   Abs   10288 #02830 -   795   639
 LNPEOL   Abs    9894 #026A6 -   560   550
=LNPEXT   Abs    9751 #02617 -   471
 LNPLB+   Abs   10012 #0271C -   627   617
 LNPLBL   Abs   10021 #02725 -   630
 LNPTR2   Abs    9788 #0263C -   499   473
 MAINLP   Ext                 -   864
 MAKEBF   Ext                 -   478
 MOVEUA   Ext                 -   893
 NOTTRS   Abs   11278 #02C0E -  2142  2133
 NTOKEN   Ext                 -  2041
 NTOKNL   Ext                 -   441  1210  1377
 NUMC++   Ext                 -  1964
 NUMCK    Ext                 -  1520  1685  1968
=Ntoken   Abs   11207 #02BC7 -  2041   783  1019  1802  1952
 OBCOLL   Ext                 -   501
 OBLCMP   Ext                 -   876
=ONP      Abs   11048 #02B28 -  1927
 ONP10    Abs   11071 #02B3F -  1945  1932
 ONP20    Abs   11091 #02B53 -  1958  1934
 ONP30    Abs   11112 #02B68 -  1964  1962
 ONP35    Abs   11125 #02B75 -  1968  1941
=ONP40    Abs   11131 #02B7B -  1974  1953
 ONP50    Abs   11156 #02B94 -  1985  1980
 ONP60    Abs   11161 #02B99 -  1990  1976  1978  1996
 ONPE1    Abs   11152 #02B90 -  1983  1966  1986  2036
 ONPE3    Abs   11087 #02B4F -  1954   708  1946  1959
 OUT1TK   Ext                 -   724
 OUT2TC   Ext                 -  1396
 OUT2TK   Ext                 -   545  1017
 OUT3TK   Ext                 -   721  1394
 OUTB+5   Ext                 -   631
 OUTBS    Abs  193935 #2F58F -    12   891
 OUTBYT   Ext                 -   779  1035  1390  1411
 OUTOVF   Ext                 -   881
 PEDIT    Ext                 -   860
 PEDITD   Ext                 -   566
 PILP+    Ext                 -  1511
=R.STPR   Abs   10701 #029CD -  1165   435
 R3=D1*   Ext                 -  1814
 RAWBFR   Abs  193920 #2F580 -    12   901
 RESPTR   Ext                 -  1412  1522  1577  1683  1834  1940  1997
=RESTRP   Abs   10834 #02A52 -  1503
=RS-R03   Abs   11283 #02C13 -  2177   700
=RSTPR1   Abs   10704 #029D0 -  1166
 RSTRP7   Abs   10872 #02A78 -  1524  1512
=RTNSET   Abs    9731 #02603 -   464   472
 S-R0-2   Abs  194683 #2F87B -    12   464  2138
 S-R0-3   Abs  194688 #2F880 -    12  2178
 S-R1-0   Abs  194689 #2F881 -    12   437
 S-R1-2   Abs  194699 #2F88B -    12   449
```

```
 S-R1-3  Abs  194704 #2F890 -    12  1166
 SAVRS+  Abs    9677 #025CD -   446
 STMTDO  Abs  194705 #2F891 -    12   458  1048
 STMTL+  Ext                -   775   826
=SVRST+  Abs    9638 #025A6 -   434
=SVRST2  Abs    9669 #025C5 -   442
 SVRSTU  Abs    9632 #025A0 -   432   569   992
 SYCOLL  Ext                -   894
 SYSFLG  Abs  194265 #2F6D9 -    12  2130
 TRANSF  Abs    9854 #0267E -   525   561
=TRNFCK  Abs   11236 #02BE4 -  2129   523   560   854
=UNPRTP  Abs   11184 #02BB0 -  2033
 UPDIN+  Ext                -   776
 UPDINA  Ext                -   432  1024
=WAITP   Abs   10862 #02A6E -  1519
=WINDWp  Abs   10950 #02AC6 -  1684
 WRDSC+  Ext                -  1674  1974
 WRDSCN  Ext                -   751  1580  1927
 WSR0-3  Ext                -   500
 XCHECK  Ext                -  1161
=XTOKCK  Abs   10688 #029C0 -  1160   443
 a'      Ext                -   575
 bSTMT   Abs    2049 #00801 -    11   878
 flRTN   Abs     -44 #FFFD4 -    11   466  2136
 lblnpe  Abs   11108 #02B64 -  1963  1991
 lnep26  Abs    9860 #02684 -   529   522
=ntken+  Abs   11204 #02BC4 -  2040   656   841
 pPARSE  Abs     244 #000F4 -    11   491
 resptr  Abs   10868 #02A74 -  1522   578   581   648  1042  1423  1517  1670
                            -  1687
 rtncc   Abs   11202 #02BC2 -  2038  1993
 sflagc  Ext                -  2137
 sflags  Ext                -   467
 t@      Ext                -   773   795
 tDISP   Ext                -  1034
 tELSE   Ext                -   752   778  1581
 tEOL    Ext                -   520   640   805  1585
 tERROR  Ext                -  1931
 tFN     Ext                -   996  1808
 tGOSUB  Ext                -  1975
 tGOTO   Ext                -  1977
 tIF     Ext                -   714
 tLBLRF  Ext                -  1410
 tLBLST  Ext                -   630
 tLINE#  Ext                -  1211
 tOFF    Ext                -  1677
 tON     Ext                -  1675
 tRESTR  Ext                -  1979
 tTIMER  Ext                -  1933
 tXWORD  Ext                -   718
```

Input Parameters

  Source file name is JP&PR1::MS

  Listing file name is JP/PR1:TI:ML::-1

  Object file name is JP%PR1:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                    STITLE WORD SCAN
                    TITLE  Parse Routines - Part 2<831212.1204>
 02C26              ABS    #02C26

*         J  PPPP    &     PPPP   RRRR    222
*         J  P   P  & &    P   P  R   R  2   2
*         J  P   P  & &    P   P  R   R      2
*         J  PPPP    &     PPPP   RRRR     22
*         J  P      & & &  P      R R     2
*      J  J  P      &  &   P      R  R    2
*       JJJ  P       && &  P      R   R  22222
*
                    RDSYMB TI%EQU::MS
*****************************************************************************
*****************************************************************************
**
** Name:(S) WRDSCN  -  Keyword Scan from Table
** Name:(S) WRDSC+  -  Keyword Scan from Table
**
** Category:   PARUTL
**
** Purpose:
**      WRDSCN tries to match the text pointed to by D1 with
**      any of the keywords specified by the caller; the
**      acceptable keyword tokens are listed in table format
**      immediately following the call to WRDSCN or WRDSC+.  If
**      one of the specified keywords is found, its corresponding
**      tokenization is output and control branches to the label
**      specified by the WRDSCN table.
**
**      To accomplish this, WRDSCN repeatedly calls NTOKEN until
**      token match is found or until all keyword tables in the
**      HP-71 have been searched.
**
**      The WRDSC+ entry point is identical to the WRDSCN entry,
**      except that WRDSC+ first calls RESPTR.
**
** Entry:
**      D(A) = (AVMEME)
**      Table address is on return stack upon entry
**      (ie. table immediately follows GOSUB.)
**      D0 points into output buffer
**
**      WRDSC+:  LEXPTR contains address pointing to optional
**               blanks preceding characters to tokenize.
**      WRDSCN:  D1 at optional blanks preceding characters
**               to tokenize.
**
** Exit:
**      P=0
**      Match found=>
**             No return to caller; control transferred to
**              specified label.
**             Token output to address pointed to by D0
**             Specified token in register A
```

```
56          **              D1 past specified keyword
57          **              D0 past keyword tokenization
58          **
59          **      Match not found=>
60          **              Return with carry clear
61          **              Last token found in A(B)
62          **              D1 past corresponding keyword
63          **
64          ** Calls:       NTOKEN,RESCAN,OUTNBS,RESPTR,XCHECK,XCHEK1
65          **
66          ** Uses.......
67          **  Exclusive: A,B,C,R1,R2
68          **  Inclusive: A,B,C,R1,R2,S0-S3,S11,R0
69          **
70          ** Stk lvls:   3
71          **
72          ** Detail:
73          **
74          **      Sample call:
75          **
76          **      GOSUBL =WRDSCN
77          **      CON(2) =tBASE           1-byte token
78          **      REL(3) =FIXP            If tBASE found, goto FIXP
79          **      CON(6) =tANGLE          3-byte token
80          **      REL(3) OPTP10           If found, goto OPTP10
81          **      CON(6) =tROUND          3-byte token
82          **      REL(3) OPTP20           If found, goto OPTP20
83          **      CON(2) 0                00 byte terminates table
84          **        ........
85          **      code continues here
86          **
87          **      ------------------------------------------
88          **
89          **      How it works:
90          **
91          **      Calls the lexical analyzer and scans through table
92          **      trying to match one of the tokens(XWORD or regular)
93          **      and jumps to an address specified in the table
94          **      table for that token.
95          **      If the token returned by the lexical analyzer is not
96          **      matched but is an XWORD, the lexical analyzer is
97          **      restarted and the table is re-scanned from the
98          **      beginning.
99          **      If no match can be found then execution continues
100         **      following the end of the table.
101         **
102         **      The table consists of any number of entries, where
103         **      each entry is a token followed by a 3-nibble relative
104         **      address which is branched to if that token is matched.
105         **      A token may be either 2 or 6 nibbles long, depending
106         **      on whether it is an XWORD/XFN/FFN token versus a
107         **      mainframe token.  The table is terminated by a 00
108         **      token; the table is immediately followed by the code
109         **      to handle the "otherwise" case (ie. the table has been
110         **      skipped over).
```

```
111             **
112             ** History:
113             **
114             **    Date      Programmer            Modification
115             **  --------    ----------   ---------------------------------
116             **  07/07/82    JP           Modified documentation
117             **  10/17/82    B.S.         Modified routine to use 3 nibble
118             **                           relative entries instead of 4 nibble
119             **                           absolute.
120             **  02/11/82    B.S.         Modified routine to handle FFNs
121             **  11/02/83    S.W.         Modified header documentation.
122             **
123             ********************************************************************
124             ********************************************************************
125 02C26 7845 =WRDSC+ GOSUB  RESPTR         Restore pointer before NTOKEN
126 02C2A 07   =WRDSCN C=RSTK
127 02C2C 109          R1=C                 Save table addr in R1
128 02C2F 7000         GOSUB  =Ntoken
129 02C33 10A   WRDS10 R2=C                 Save Lex restart info
130 02C36 136          CDOEX
131 02C39 06           RSTK=C               Save output pointer on stack
132 02C3B 119          C=R1                 Recall table address
133 02C3E 134          DO=C                 Point DO into table
134 02C41 20    WRDS20 P=     0
135 02C43 15E5         C=DATO 6             Read token from table
136 02C47 AF5          B=C    W             Save table entry in B
137 02C4A 3100         LC(2)  =tXWORD       Yes, then check if restart needed
138 02C4E 96D          ?B#0   B             Is this the end of the table
139 02C51 82           GOYES  WRDS50        No, then check this entry
140 02C53 7170         GOSUB  XCHEK1
141 02C57 4E0          GOC    WRDS40        XWORD,XFN, or funny FN ?
142 02C5A 161          DO=DO+ 2             No, then no match found
143 02C5D 07           C=RSTK               Recall output pointer
144 02C5F 136          CDOEX                DO=output pointer, C is continue ad
145 02C62 06           RSTK=C               Push return address on stack
146 02C64 03    WRDS35 RTNCC                Jump to address past table
147             *-
148             *-
149 02C66 07    WRDS40 C=RSTK               C <-- Output pointer
150 02C68 7605         GOSUB  RESPTR        Restore original input pointer
151 02C6C 112          A=R2                 Recall lex restart info
152 02C6F 8E00         GOSUBL =RESCAN       Restart lexical analyzer
          00
153 02C75 6DBF         GOTO   WRDS10        Re-scan table
154             *-
155             *-
156 02C79 25    WRDS50 P=     5             Length=6
157 02C7B 961          ?B=C   B             Is it an XWORD?
158 02C7E 81           GOYES  WRDS55        Yes, then skip
159 02C80 20           P=     0
160 02C82 3100         LC(2)  =tXFN
161 02C86 25           P=     5
162 02C88 961          ?B=C   B             Is it an XFN ?
163 02C8B B0           GOYES  WRDS55        Yes, then skip
164 02C8D E6           C=C+1  A
```

```
165 02C8F 961          ?B=C   B            Is it a FFN ?
166 02C92 40           GOYES  WRDS55        Yes, then skip
167 02C94 21           P=     1            No, then length=2
168 02C96 136   WRDS55 CDOEX
169 02C99 809          C+P+1               Skip past token
170 02C9C 136          CDOEX
171 02C9F 162          DO=DO+ 3            Skip past jump address
172 02CA2 914          ?B#A   WP           Is it a match?
173 02CA5 C9           GOYES  WRDS20        No, then continue search
174 02CA7 182          DO=DO- 3            Yes, then back up to jump address
175 02CAA 07           C=RSTK              Recall output pointer
176 02CAC 136          CDOEX               DO=Output pointer,C(A)=table pointe
177 02CAF 06           RSTK=C              Save table pointer on stack
178 02CB1 8E00         GOSUBL =OUTNBS      Output token
          00
179 02CB7 07           C=RSTK              Recall table pointer
180 02CB9 136          CDOEX               DO=Table pointer,C(A)=Output pointe
181 02CBC 06           RSTK=C              Save Output pointer on RSTK for REL
182 02CBE 8C00         GOLONG =REL3DO      Jump to address pointed to by DO
          00
183             *
184 02CC4 3100 =XCHECK LC(2)  =tXWORD
185 02CC8 962   XCHEK1 ?A=C   B
186 02CCB 00           RTNYES
187 02CCD 3100         LC(2)  =tXFN
188 02CD1 962          ?A=C   B
189 02CD4 00           RTNYES
190 02CD6 E6           C=C+1  A
191 02CD8 962          ?A=C   B            funny FN ?
192 02CDB 00           RTNYES
193 02CDD 01           RTN                 Carry clr => none of the above
194             *
```

```
                STITLE Output Token(s) Routines
*****************************************************************
*****************************************************************
**
** Name:(S) OUT1TK  -  Output 1 byte from A(B)
** Name:(S) OUT1T+  -  Increment D1, Output 1 byte from A(B)
** Name:(S) OUTBYT  -  Output 1 byte from C(B)
** Name:(S) OUTBY+  -  Increment D1, Output 1 byte from C(B)
** Name:(S) OUT2TK  -  Output 2 bytes from A(3-0)
** Name:(S) OUT2TC  -  Output 2 bytes from C(3-0)
** Name:(S) OUT3TK  -  Output 3 bytes from A(5-0)
** Name:(S) OUT3TC  -  Output 3 bytes from C(5-0)
** Name:(S) OUTNIB  -  Output 1 nibble from C(0)
**
** Category:   GENUTL
**
** Purpose:
**      Output specified number of nibbles to address pointed
**      to by D0; a check is made so that D0 does not write
**      past available memory end.
**
** Entry:
**      D(A) = (AVMEME) - Available Memory End
**      D0   = address at which output to go
**
**      OUTNIB:  Nibble to output in C(0)
**      OUT1TK:  Byte to be output in A(B)
**      OUT1T+:  Byte to be output in A(B)
**      OUTBYT:  Byte to be output in C(B)
**      OUTBY+:  Byte to be output in C(B)
**      OUT2TK:  2 Bytes to be output in A(3-0)
**      OUT2TC:  2 Bytes to be output in C(3-0)
**      OUT3TK:  3 bytes to be output in A(5-0)
**      OUT3TC:  3 bytes to be output in C(5-0)
**
** Exit:
**      No memory error =>
**        Carry clear on exit
**        D0 incremented past the tokens that were output
**        D1 incremented by 2 (OUT1T+, OUTBY+ entries only)
**        A(B) & C(B) are swapped (OUTBYT, OUTBY+ entry)
**        A(A) & C(A) are swapped (OUT2TC entry only)
**        A(W) & C(W) are swapped (OUT3TC entry only)
**
**      Else
**        golong MEMERR
**
** Calls:       OVFLCK
**
** Uses:        D0                  (OUTNIB,OUT1TK,OUT2TK,OUT3TK)
**              D1,D0                       (OUT1T+)
**              A(B),C(B), D0               (OUTBYT)
**              A(B),C(B), D1,D0            (OUTBY+)
**              A(A),C(A), D0               (OUT2TC)
**              A,C, D0                     (OUT3TC)
```

```
250                **
251                ** Stk lvls:   1
252                **
253                ** History:
254                **
255                **    Date      Programmer              Modification
256                **  --------    ----------   ----------------------------------
257                **  07/07/82    JP           Modified documentation
258                **  11/02/83    S.W.         Modified documentation header.
259                **
260                ********************************************************************
261                ********************************************************************
262                *
263 02CDF 171     =OUT1T+ D1=D1+ 2
264 02CE2 580             GONC   OUT1TK        (B.E.T.)
265                *
266 02CE5 171     =OUTBY+ D1=D1+ 2
267 02CE8 AEE     =OUTBYT ACEX   B
268 02CEB 161     =OUT1TK DO=DO+ 2
269 02CEE 7940            GOSUB  OVFLCK
270 02CF2 181             DO=DO- 2
271 02CF5 148             DATO=A B
272 02CF8 161             DO=DO+ 2
273 02CFB 03              RTNCC
274 02CFD DE      =OUT2TC ACEX   A
275 02CFF 163     =OUT2TK DO=DO+ 4
276 02D02 7530            GOSUB  OVFLCK
277 02D06 183             DO=DO- 4
278 02D09 1583            DATO=A 4
279 02D0D 163             DO=DO+ 4
280 02D10 03              RTNCC
281 02D12 AFA     =OUT3TC A=C    W
282 02D15 165     =OUT3TK DO=DO+ 6
283 02D18 7F10            GOSUB  OVFLCK
284 02D1C 185             DO=DO- 6
285 02D1F 1585            DATO=A 6
286 02D23 165             DO=DO+ 6
287 02D26 03              RTNCC
288 02D28 160     =OUTNIB DO=DO+ 1
289 02D2B 7C00            GOSUB  OVFLCK
290 02D2F 180             DO=DO- 1
291 02D32 15C0            DATO=C 1
292 02D36 160             DO=DO+ 1
293 02D39 03              RTNCC
```

```
                        STITLE OVFLCK - Overflow Check
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:    OVFLCK  -  Memory Overflow Check
                    **
                    ** Category:   GENUTL
                    **
                    ** Purpose:
                    **      Ensure that D0 setting is not beyond the end of
                    **      available memory.
                    **
                    ** Entry:
                    **      D(A)   =  (AVMEME)
                    **      D0 contains proposed end of output buffer
                    **
                    ** Exit:
                    **        Return to caller with carry clear if D0 <= D(A)
                    **        Else golong MEMERR
                    **
                    ** Calls:      None
                    **
                    ** Uses....... None
                    **
                    ** Stk lvls:   0
                    **
                    **    Date      Programmer              Modification
                    **   --------   ----------   ----------------------------------
                    **  07/07/82   JP           Modified documentation
                    **
                    ****************************************************************
                    ****************************************************************
02D3B 136    OVFLCK CD0EX
02D3E 8B7           ?C>D   A
02D41 70            GOYES  OUTOVF
02D43 136           CD0EX
02D46 01            RTN
02D48 8C00  =OUTOVF GOLONG =MEMERR
      00
```

```
                        STITLE OPTION Parse
                ********************************************************************
                ********************************************************************
                **
                ** Name:    OPTP    -  OPTION Statement Parse
                **
                ** Category:   STPARS
                **
                ** Purpose:
                **      Parse OPTION statment
                **
                ** Entry:
                **      D1 past OPTION keyword
                **      D0 past tOPT'N
                **      D(A) = (AVMEME)
                **
                ** Exit:
                **      Valid statement parse =>
                **        Return with carry clear
                **        D1 past syntactically correct statement
                **        D0 past tokenized statement in output buffer
                **        P=0
                **      Exit to PARERR if parse error
                **
                ** Calls:      WRDSCN
                **
                ** Uses.......
                **  Exclusive: A,C, D0,D1
                **  Inclusive: A-C, D0,D1, R0-R2, S0-S3,S11
                **
                ** Stk lvls:   4
                **
                ** Detail:
                **      OPTION BASE <numeric expression>
                **      OPTION ANGLE  DEGREES | RADIANS
                **      OPTION ROUND  NEAR | PINF | NINF | ZERO
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ----------------------------------
                **  07/07/82   JP           Modified documentation
                **  10/06/82   PM           Added Round option
                **
                ********************************************************************
                ********************************************************************
                *
02D4E 78DE =OPTP   GOSUB  WRDSCN
02D52 00           CON(2) =tBASE
02D54 000          REL(3) =FIXP          NUMCK, RESPTR
02D57 3B10         CON(6) =tANGLE
          60
02D5D 110          REL(3) OPTP10
02D60 FE10         CON(6) =tROUND
          C4
```

```
385 02D66 B10          REL(3) OPTP20
386 02D69 00           CON(2) 0
387 02D6B 5F3          GONC   OPTP30        (B.E.T.) - Ill. Parm error
388               *
389 02D6E 78BE OPTP10  GOSUB  WRDSCN
390 02D72 00           CON(2) =tDEGRE
391 02D74 746          REL(3) RTNCC
392 02D77 00           CON(2) =tRDIAN
393 02D79 246          REL(3) RTNCC
394 02D7C 00           CON(2) 0
395 02D7E 5C2          GONC   OPTP30        (B.E.T.)
396               *
397 02D81 75AE OPTP20  GOSUB  WRDSCN
398 02D85 FE10         CON(6) =tNEAR
          C3
399 02D8B 036          REL(3) RTNCC
400 02D8E 3B10         CON(6) =tPOS
          24
401 02D94 726          REL(3) RTNCC
402 02D97 FE10         CON(6) =tNEG
          D3
403 02D9D E16          REL(3) RTNCC
404 02DA0 FE10         CON(6) =tZERO
          C1
405 02DA6 516          REL(3) RTNCC
406 02DA9 00           CON(2) 0
407 02DAB 6390 OPTP30  GOTO   ERR03
```

```
                        STITLE Update INADDR
                *****************************************************************
                *****************************************************************
                **
                ** Name:    UPDINA  -  Update INADDR, Step over Stmt Length
                **
                ** Category:   PARUTL
                **
                ** Purpose:
                **      Updates INADDR & steps over stmt length byte
                **
                ** Entry:
                **      UPDINA - D0 points at statement length byte
                **      UPDIN+ - C(A) points at statement length byte
                **
                ** Exit:
                **      INADDR updated, and D0 incremented past stmt length
                **      Carry clear
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Exclusive: INADDR,D0, UPDINA: C(A)
                **  Inclusive: INADDR,D0, UPDINA: C(A)
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------   ----------   ---------------------------------
                **  07/07/82   JP           Modified documentation
                **
                *****************************************************************
                *****************************************************************
02DAF 136       =UPDINA CDOEX
                *
02DB2 1B00      =UPDIN+ D0=(5) =INADDR
      000
02DB9 144               DAT0=C A
02DBC 134               D0=C
02DBF 161               D0=D0+ 2
02DC2 01                RTN
```

```
450                     STITLE Calculate Statement Length
451                 ********************************************************************
452                 ********************************************************************
453                 **
454                 ** Name:    STMTLO  -  Calculate and Write out Statement Length
455                 ** Name:    STMTL+  -  Calculate and Write out Statement Length
456                 **
457                 ** Category:   PARUTL
458                 **
459                 ** Purpose:
460                 **      Calculates and writes out statement length.
461                 **      STMTL+ also clears S10, the implied LET error flag.
462                 **
463                 ** Entry:
464                 **      DO past tokenized statement terminator
465                 **      INADDR points to statement length byte
466                 **
467                 ** Exit:
468                 **      If tokenized statement <= 253 nibbles
469                 **       Exit with carry clear
470                 **       Statement length byte written out to INADDR.
471                 **       (S-R0-0) = (INADDR) - For literal labels in an IF stmt
472                 **
473                 **      Else error exit to MFERR
474                 **
475                 ** Calls:      OUTBYT
476                 **
477                 ** Uses.......
478                 **  Exclusive: A(A),C(A),R0,S-R0-0, STMTL: S10
479                 **  Inclusive: A(A),C(A),R0,S-R0-0, STMTL: S10
480                 **
481                 ** Stk lvls:   2
482                 **
483                 ** Detail:
484                 **      Doesn't update INADDR
485                 **
486                 ** History:
487                 **
488                 **    Date     Programmer              Modification
489                 **  --------   ----------   ----------------------------------
490                 **  07/07/82   JP           Modified documentation
491                 **  04/27/83   S.W.         Added 'Line Too Long' error exit
492                 **  05/05/83   S.W.         Limit length of tokenized stmt to FD
493                 **                           to fix restart after THEN bug#649-4
494                 **
495                 ********************************************************************
496                 ********************************************************************
497                 *
498                 * Calculate Statement Length
499                 *
500 02DC4 84A   =STMTL+ ST=0   10              Clear Implied LET error flag
501 02DC7 7D1F =STMTLO GOSUB  OUTBYT
502 02DCB 136         CDOEX                 Output Buffer End
503 02DCE 108         R0=C
504 02DD1 1B00        D0=(5) =INADDR
```

```
          000
505 02DD8 142          A=DAT0 A              ADDR OF STATEMENT START
506 02DDB 86A          ?ST=0  10
507 02DDE B0           GOYES  STMTL2
508 02DE0 1A00         D0=(4) =S-R0-0
          00
509 02DE6 140          DAT0=A A              Save INADDR in S-R0-0
510 02DE9 E2    STMTL2 C=C-A  A              LENGTH OF STATEMENT
511 02DEB 92E          ?C#0   XS
512 02DEE 41           GOYES  STMTLe         Tokenized stmt > 253 nibs ?
513 02DF0 CE           C=C-1  A
514 02DF2 CE           C=C-1  A
515 02DF4 130          D0=A
516 02DF7 14C          DAT0=C B              WRITE OUT STATEMENT LENGTH
517 02DFA 118          C=R0
518 02DFD 134          D0=C
519 02E00 01           RTN
520             *
521             * Even if XWORD or XFN, don't bother restarting since parse was 1
522             *
523 02E02 1A00  STMTLe D0=(4) =eL2LNG
          00
524 02E08 6741         GOTO   ERRX20
525             *
```

```
                        STITLE ! Check
                ********************************************************************
                ********************************************************************
                **
                ** Name:     !CK     -  Check for "!"; if found, parses remark
                ** Name:     !CK2    -  Check for "!"; if found, parses remark
                ** Name:     !CK3    -  Outputs "!", then parses remark
                **
                ** Category:   PARUTL
                **
                ** Purpose:
                **      Checks for !
                **
                ** Entry:
                **      P=0
                **      D0 points into output buffer
                **      D(A) = (AVMEME)
                **      3 entry points:
                **      1) !CK  - LEXPTR contains address pointing to alleged !
                **      2) !CK2 - D1 points at alleged !
                **      3) !CK3 - D1 past !
                **
                ** Exit:
                **      For entry points !CK and !CK2 :
                **      Carry set => ! not found
                **                   D1=(LEXPTR)  - !CK entry only
                **      Carry clr => comment parsed
                **                   D1 at 0D (end of line)
                **                   D0 past tokenized comment
                **
                **      For entry point !CK3 :
                **      See carry clr above
                **
                **      All three entry points can conceivably error exit
                **      with MEMERR - Insufficient memory to tokenize line
                **
                ** Calls:      RESPTR, OUTBYT
                **
                ** Uses.......
                **  Exclusive: A,C,D0,D1
                **
                ** Stk lvls:   2
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ----------------------------------
                **  07/07/82   JP           Modified documentation
                **
                ********************************************************************
                ********************************************************************
                *
                *
02E0C 7263      =!CK    GOSUB  RESPTR
02E10 14B               A=DAT1 B
```

```
581 02E13 3112 =!CK2   LCASC  \!\
582 02E17 966          ?A#C   B
583 02E1A 00           RTNYES
584             *
585 02E1C 171   !CK+   D1=D1+ 2
586             *
587 02E1F 3100 =!CK3   LC(2)  =t!
588 02E23 71CE         GOSUB  OUTBYT
589 02E27 6000         GOTO   =REMP          SUPPRESS 1ST LEADING BLANK
```

```
590                 STITLE Parse Error Exit Routine
591             ********************************************************************
592             ********************************************************************
593             **
594             ** Name:(S) SYNTXe  -  "Syntax" Parse Error Exit
595             ** Name:(S) IVEXPe  -  "Invalid Expression" Parse Error Exit
596             ** Name:(S) IVPARe  -  "Invalid Parameter" Parse Error Exit
597             ** Name:    ERR3    -  "Invalid Parameter" Parse Error Exit
598             ** Name:(S) MSPARe  -  "Missing Parameter" Parse Error Exit
599             ** Name:(S) IVVARe  -  "Invalid Variable" Parse Error Exit
600             ** Name:(S) ILCNTe  -  "Illegal Context" Parse Error Exit
601             ** Name:(S) EXCHRe  -  "Excess Characters" Parse Error Exit
602             ** Name:(S) QUOEXe  -  "Quote Expected" Parse Error Exit
603             ** Name:(S) PRNEXe  -  ") Expected" Parse Error Exit
604             ** Name:(S) FSPECe  -  "Invalid Filespec" Parse Error Exit
605             ** Name:(S) PARERR  -  Generic Parse Error Exit
606             **
607             ** Category:   PARUTL
608             **
609             ** Purpose:
610             **      Parse ERROR Exit Routines.
611             **      The 1st 11 entry points above all fall into PARERR.
612             **      Depending on entry conditions, PARERR may:
613             **      1) Display the error message and redisplay the line,
614             **         with the cursor flashing on the character pointed
615             **         to by D1 or (LEXPTR).
616             **      or
617             **      2) Attempt to reparse the statement as an implied
618             **         DISP. (S10=1,S9=0 on entry)
619             **      or
620             **      3) Attempt to reparse the statement as an implied
621             **         GOTO <label>. (S9=S10=1 on entry)
622             **      or
623             **      4) Restart the lexical analyzer and reparse the
624             **         entire statement. (RESTART flag nonzero)
625             **
626             ** Entry:
627             **      S4=1 if D1 set at error position.
628             **      S4=0 if LEXPTR contains address of error position.
629             **      S10=1 if implied LET error (try implied DISP)
630             **      S10=S9=1 if middle of IF stmt and implied LET error
631             **
632             **      This entry condition is handled by the driver:
633             **      S-R1-3 (RESTART Flag) = 0 => Don't restart
634             **                            = F => Normal restart
635             **                            = E => Restart of extended IF
636             **
637             **      PARERR - Lower 4 nibbles of D0 contain error#
638             **
639             ** Exit:
640             **      If S10=0, (S-R1-3)=0 on entry
641             **         Exit through MFERR:
642             **         Display error message
643             **         Redisplay Input Line with Cursor at Error
644             **         Returns to Main Loop
```

```
645             **
646             **      If RESTART flag set (S-R1-3)#0 on entry
647             **         exit through RESTAR
648             **
649             **      If 'Normal' implied LET error  (S10=1 & S9=0)
650             **         Try implied DISP parse
651             **
652             **      If Implied LET error & Middle of IF (S10=S9=1)
653             **         Try implied GOTO <label> parse
654             **
655             ** Calls:       RESPTR, R3=D10, D1C=R3, EOLCK, RSTRT?, TRNFCK,
656             **              EOLCK+, NTKEN+, UPDIN+, LBLINP
657             **
658             ** Uses:        A-C, R0,R3, D0,D1, S4,S8-S10
659             **
660             ** Stk lvls:    1
661             **              6 if Implied LET/Middle of IF/Restart
662             **
663             **
664             ** Algorithm:
665             **      If S4=0
666             **        THEN RESPTR
667             **      If RESTART flag (S-R1-3) set
668             **        THEN goto RESTAR;
669             **        ELSE If previously restarted (S-R1-0 [err#] #0)
670             **               THEN Restore D1 to original error position
671             **                    using S-R1-1; Set D0 from S-R1-0;
672             **             If Implied LET error (S10=1)
673             **               Restore D1,D0 from R3; Clear S10;
674             **               If not in middle of IF (S9=0)
675             **                 THEN try implied DISP
676             **                 ELSE Decrement D0 4 nibbles
677             **                      (over tEXTIF & stmt length byte);
678             **                      Recover old INADDR from S-R0-0;
679             **                      Call GOSUBP;
680             **             Handle as error.
681             **
682             ** Note:
683             **      If error is ILLEGAL CONTEXT & S9 is set, then S10
684             **      is cleared.  This prevents illegal context errors
685             **      immediately after THEN/ELSE from being interpreted
686             **      as labels.
687             **
688             ** History:
689             **
690             **    Date      Programmer   Modification
691             **  --------    ----------   -------------
692             **  01/07/83    S.W.         Added algorithm
693             **  02/04/83    JP           Added mneumonc entry point names
694             **
695             ********************************************************************
696             ********************************************************************
697             *
698 02E2B      =SYNTXe
699 02E2B 1A00 =ERR01  D0=(4) =eSYNTX        Syntax
```

```
          00
700 02E31 66D0          GOTO   ERRX0
701               *
702 02E35       =IVEXPe
703 02E35 1A00  =ERR02  D0=(4) =eILEXP        Invalid Expression
          00
704 02E3B 6CC0          GOTO   ERRX0
705               *
706 02E3F       =IVPARe                       Invalid Parameter
707 02E3F 874   =ERR03  ?ST=1  4              NO RESTORE OF INPUT PTR?
708 02E42 60            GOYES  ERR3*
709 02E44 7A23          GOSUB  RESPTR         MUST CALL WRDSCN 'CAUSE OF tELSE
710 02E48 7000   ERR3*  GOSUB  =EOLCK+        Check if @ EOL
711 02E4C 844           ST=0   4              NEED INPUT PTR RESTORED NOW
712 02E4F 4C0           GOC    ERR04
713               *
714               *
715 02E52 1A00  =ERR3   D0=(4) =eILPAR        Invalid Param
          00
716 02E58 6FA0          GOTO   ERRX0
717               *
718 02E5C       =MSPARe
719 02E5C 1A00  =ERR04  D0=(4) =eMSPAR        Missing Param
          00
720 02E62 65A0          GOTO   ERRX0
721               *
722 02E66       =IVVARe
723 02E66 1A00  =ERR05  D0=(4) =eILVAR        Invalid Var
          00
724 02E6C 6B90          GOTO   ERRX0
725               *
726 02E70       =ILCNTe
727 02E70 1A00  =ERR06  D0=(4) =eILCNT        Illegal context
          00
728 02E76 869           ?ST=0  9
729 02E79 E0            GOYES  errx0
730               * If S10 & S9 set, then ILLEGAL CONTEXT after THEN/ELSE
731               *  Don't want it interpreted as a label
732 02E7B 84A           ST=0   10
733 02E7E 580           GONC   errx0          (B.E.T.)
734               *
735 02E81       =EXCHRe
736 02E81 1A00  =ERR08  D0=(4) =eEXCHR        Excess Char
          00
737 02E87 6080   errx0  GOTO   ERRX0
738               *
739 02E8B       =QUOEXe
740 02E8B 1A00  =ERR09  D0=(4) =eQUOEX        Quote Expected
          00
741 02E91 6670          GOTO   ERRX0
742
743 02E95       =PRNEXe
744 02E95 1A00  =ERR10  D0=(4) =ePRNEX        ) Expected
          00
745 02E9B 6C60          GOTO   ERRX0
```

```
746                *
747                * Implied LET Error AND Middle of IF (ERRIF* entry)
748                *  OR
749                * Failed parse of restarted keyword (ERRIF entry)
750                *  (may be function or some other keyword)
751                *
752                * LBLINP clears S9,S10
753                *
754                * Don't allow unquoted label that contains a begin BASIC
755                *  keyword, whose parse previously failed - restore error info
756                *
757 02E9F 854    ERRIF  ST=1   4              Flag for don't allow unquoted lbl
758 02EA2 7000         GOSUB  =ntken+         containing begin BASIC keyword
759 02EA6 873          ?ST=1  3
760 02EA9 50           GOYES  ERRIF1         BEGIN BASIC?
761 02EAB 844          ST=0   4              Allow unquoted label
762                * Don't want failed Begin BASIC XWORD parse to be interpreted
763                *  as a label - want to see the correct parse error
764 02EAE 70C2 ERRIF1 GOSUB  RESPTR
765 02EB2 183  ERRIF* D0=D0- 4              Back up over tEXTIF & STMT LEN
766 02EB5 132          ADOEX                 SAVE D0
767 02EB8 1B00         D0=(5) =S-R0-0
          000
768 02EBF 146          C=DAT0 A              Get old INADDR (before tIF)
769 02EC2 7CEE         GOSUB  UPDIN+
770 02EC6 130          D0=A                  RESTORE D0
771 02EC9 7000         GOSUB  =LBLINP
772 02ECD 471          GOC    ERRIF2         Error in Label? (Syntax error)
773                * Legal label
774 02ED0 877          ?ST=1  7              String expr?
775 02ED3 70           GOYES  ERRIF+
776                * Unquoted label
777 02ED5 874          ?ST=1  4              Failed begin BASIC keyword parse?
778 02ED8 02           GOYES  ERRIF9          Yes=> RESTART or restore error inf
779 02EDA 7000 ERRIF+ GOSUB  =EOLCK
780 02EDE 5F0          GONC   ERRIF3         Excess characters? - N.Func w/parms
781 02EE1 6000         GOTO   =LNP55         Restore ptr; goto LNEP66
782                * S9,S10 are clear; error in label parse
783 02EE5 1A00 ERRIF2 D0=(4) =eSYNTX        1st char illegal
          00
784 02EEB 4C0          GOC    ERRIF9         (B.E.T.)
785                *
786 02EEE 1A00 ERRIF3 D0=(4) =eEXCHR        Excess characters
          00
787 02EF4 7A72 ERRIF7 GOSUB  RESPTR
788                *
789 02EF8 858  ERRIF9 ST=1   8
790 02EFB 6810         GOTO   ERRX11
791                *
792 02EFF 4F9  errif   GOC    ERRIF          (B.E.T.)
793                *
794 02F02      =FSPECe
795 02F02 1A00 =ERR11 D0=(4) =eFSPEC        Illegal Filespec
          00
796                *
```

```
    797             * ERRX0 Entry
    798             *
    799             * D0 = Error Number
    800             *
    801             * If S4=0 then Restore Input Pointer
    802             *
    803 02F08       =PARERR
    804 02F08 874   =ERRX0  ?ST=1  4
    805 02F0B 60            GOYES  ERRX10
    806 02F0D 7162          GOSUB  RESPTR
    807             *
    808 02F11 848   =ERRX10 ST=0   8
    809 02F14 844    ERRX11 ST=0   4
    810 02F17 7DE0          GOSUB  RSTRT?
    811 02F1B 585           GONC   RESTAR          Restart?
    812 02F1E 20            P=     0
    813 02F20 86A           ?ST=0  10              Not implied LET error?
    814 02F23 D2            GOYES  ERRX20
    815 02F25 7E11          GOSUB  D1C=R3
    816 02F29 134           D0=C
    817 02F2C 84A           ST=0   10              Clear Implied LET error
    818 02F2F 879           ?ST=1  9
    819 02F32 08            GOYES  ERRIF*
    820             * Implied LET error
    821             * If in a TRANSFORM, DO NOT allow Implied DISP
    822 02F34 7000          GOSUB  =TRNFCK
    823 02F38 5E2           GONC   ERRIDS
    824             *
    825 02F3B 87A   =ERRDSP ?ST=1  10              Traps out string functions
    826 02F3E 1C            GOYES  errif            after THEN/ELSE
    827 02F40 3100          LC(2)  =tDISP
    828 02F44 70AD          GOSUB  OUTBYT
    829 02F48 7000          GOSUB  =DISPP
    830             *
    831 02F4C 6000   lnep66 GOTO   =LNEP66
    832             *
    833             * Set P to indicate Parse Error, Update ERR#
    834             * Set A=0 to get BASIC prompt
    835             * gosub to Mainframe Error Handler  (MFERR-)
    836             * golong Main Loop
    837             *
    838             *  If called by TRANSFORM, exit with D0 loaded
    839             *  with error#, and carry set
    840             *
    841 02F50 7000 =ERRX20 GOSUB  =TRNFCK
    842 02F54 581           GONC   ERR-TR
    843             *
    844 02F57 28            P=     8               Parse Error setting
    845 02F59 D0            A=0    A               BASIC cursor setting
    846 02F5B 8E00          GOSUBL =MFERR-         Assumes D0=Error #
              00
    847 02F61 8C00          GOLONG =MAIN10         Return to Main loop
              00
    848             *
    849 02F67 1A00   ERRIDS D0=(4) =eSYNTX
```

```
            00
    850 02F6D 06    ERR-TR RSTK=C              Push address on stack
    851 02F6F 13E          CDOXS               Want err# in C(3-0)
    852 02F72 02           RTNSC
    853             *
```

```
854                 STITLE RESTAR - Restart Lex Analyzer
855               ********************************************************************
856               ********************************************************************
857               **
858               ** Name:    RESTAR  -  Restart Lex Analyzer
859               ** Name:(S) REST*   -  Restart Lex Analyzer
860               **
861               ** Category:   PARUTL
862               **
863               ** Purpose:
864               **      Restarts the Lexical Analyzer when the parse of an XWORD
865               **      token fails; allows the parser to find smaller keywords i
866               **      the same LEX file, as well as similarly spelled keywords
867               **      in other LEX files.
868               **
869               **      The RESTAR entry point is used by the parse error driver
870               **      to try all possible statement parses, before reporting an
871               **      error; the original parse error and position is saved and
872               **      is later restored if all subsequent parse attempts fail.
873               **
874               **      The REST* entry point is used by a LEX file when a parse
875               **      fails and it is known that RESTAR will find a subsequent
876               **      statement parse in the mainframe which can give a clearer
877               **      more coherent error message.  This entry point ensures
878               **      that the caller's error number and error position is NOT
879               **      preserved anywhere - it is as though the keyword was neve
880               **      found.
881               **
882               ** Entry:
883               **       (STMTDO) = Input pointer for restart
884               **       (S-R1-2) = Restart Address
885               **
886               **      2 entry points:
887               **      RESTAR - If RESTAR hasn't been previously called
888               **                Then C(A)=0
889               **                     D0=Latest error# generated
890               **                     D1=S-R1-0
891               **                     A(A)=Error position
892               **                Else...
893               **                     (S-R1-0)=Original error#
894               **                     (S-R1-1)=Original error position
895               **               If not failed label parse after THEN/ELSE
896               **                Then S8=0
897               **                     (INADDR) = addr of last stmt length byte
898               **                     C(S)#E iff Extended IF
899               **                Else...
900               **                     S8=1
901               **                     R3(A) pts 2 nibs past last stmt len byte
902               **
903               **
904               **      REST*  -  (INADDR) = address of last stmt length byte
905               **                (S-R1-3)#F iff Extended IF
906               **
907               **
908               ** Exit:
```

```
909             **      Control is turned over to the main parse driver.
910             **
911             ** Calls:      RESPTR,RESCAN,R.STPR,STLXP2,SVRST2,EXTIF+
912             **
913             ** Uses.......
914             **  Exclusive: A,C,D1,D0,S8
915             **  Inclusive: A-C, D1,D0, S0-S3,S8,S11, R0,R3
916             **
917             ** Stk lvls:   3
918             **
919             ** Detail:
920             **      The component parts of RESTART are as follows:
921             **      S-R1-0  Original error#; set prior to 1st time
922             **                through RESTAR
923             **      S-R1-1  Original error position; set prior to 1st
924             **                time through RESTAR
925             **      S-R1-3  Flags the parse error handler whether or
926             **              not to RESTART the lexical analyzer.
927             **              If S-R1-3 is nonzero, STMTD0 contains the
928             **              address at which to set D1 to restart and
929             **              S-R1-2 contains the restart address.
930             **              S-R1-3 is cleared when NTOKEN is first
931             **              called; It is set (along with STMTD0) when
932             **              the begin BASIC token is an XWORD.
933             **      S-R1-2  Contains RESTART address.  Set initially
934             **              when NTOKEN first called.  Updated when
935             **              RESCAN called in RESTAR.
936             **      STMTD0  Contains address at which D1 should be at
937             **              when restarting the lexical analyzer.  Set
938             **              and cleared with S-R1-3.
939             **
940             ** Algorithm:
941             **      If 1st time thru RESTART for this lexeme
942             **      (S-R1-0 contains 0)
943             **        Save err# in S-R1-0 & position in S-R1-1;
944             **      Clear RESTART flag (S-R1-3);
945             **      Get input ptr from STMTD0 & write out to LEXPTR
946             **      (needed 'cause RESCAN doesn't save as NTOKEN does);
947             **      Retrieve RESTART addr for lexical analyzer (S-R1-2);
948             **      Restore D0 from INADDR;
949             **      Call RESCAN; Set RESTART flag (S-R1-3) if XWORD/XFN;
950             **      Save RESTART address in S-R1-2;
951             **      Goto H (main parse driver - JP&PR1).
952             **
953             ** History:
954             **
955             **    Date      Programmer          Modification
956             **  --------    ----------  ----------------------------------
957             **  07/06/82    JP          Modified documentation
958             **  08/23/82    S.W.        Added documentation on S-R1-2, S-R1-3
959             **                           and STMTD0.
960             **  11/15/82    S.W.        Deleted error exit option - wasn't
961             **                           used anywhere
962             **  05/24/83    S.W.        Added REST* entry point for use by
963             **                          language extensions; this is an alter
```

```
 964                **                          tive to the 'usual' error exit (the
 965                **                          usual error exit saves the original
 966                **                          error and restores it if no other par
 967                **                          works).  REST* can be used ONLY if it
 968                **                          known that restart will eventually gi
 969                **                          control to a mainframe parse routine;
 970                **                          REST* can be useful to prevent obscur
 971                **                          error messages.  If a previous parse
 972                **                          error occured, the first one generate
 973                **                          in the 'usual' way is preserved; othe
 974                **                          wise the next error generated in the
 975                **                          'usual' way (not using REST*) is
 976                **                          preserved.
 977                **                          For example:
 978                **                           The HPIL parse for ON INTR, may choo
 979                **                          to suppress its error message/positio
 980                **                          in favor of any one given by ON ERROR
 981                **                          TIMER|<expr>
 982                **
 983                ************************************************************
 984                ************************************************************
 985                * Attempt to RESTART
 986                * If 1st time through, save err# & err position
 987 02F74 8AE     RESTAR ?C#0   A              Not 1st time through?
 988 02F77 E0           GOYES  RESTA1
 989 02F79 136          CDOEX                  Err#
 990 02F7C 145          DAT1=C A
 991 02F7F 174          D1=D1+ 5
 992 02F82 141          DAT1=A A               Error position
 993                * NONE FOUND
 994                * Error# & Error position in S-R1-0 & S-R1-1
 995                *
 996 02F85 878     RESTA1 ?ST=1  8              Failed Label parse after THEN?
 997 02F88 72           GOYES  RESTA3
 998                * Put output ptr in R3 (2 nibs after where output to go)
 999 02F8A 1B00    RESTxx D0=(5) =INADDR
           000
1000 02F91 146          C=DAT0 A               Address of last stmt length byte
1001 02F94 858          ST=1   8               Immediately after THEN/ELSE
1002 02F97 B46          C=C+1  S
1003 02F9A B46          C=C+1  S
1004 02F9D 5E0          GONC   RESTA2          EXTENDED IF? INADDR past tEXITF
1005 02FA0 134          D0=C                   2 nibs PRIOR to where output to go
1006 02FA3 163          D0=D0+ 4               Position properly
1007 02FA6 136          CDOEX
1008 02FA9 848          ST=0   8               Not immediately after THEN/ELSE
1009 02FAC 10B    RESTA2 R3=C                   Output pointer
1010 02FAF AC0    RESTA3 A=0    S
1011 02FB2 7000         GOSUB  =R.STPR         Clear RESTART flag
1012 02FB6 1F00         D1=(5) =STMTD0
           000
1013 02FBD 147          C=DAT1 A               Input ptr for LEX restart
1014 02FC0 1C5          D1=D1- 6
1015 02FC3 143          A=DAT1 A               RESTART ADDR IN A
1016               * Since RESCAN doesn't set LEXPTR as NTOKEN does
```

```
1017 02FC6 8E00          GOSUBL =STLXP2        Writes out C to LEXPTR
           00
1018               *
1019               * Restore D1 - Input buffer position before XWORD parse
1020               * Compute Output buffer position before XWORD
1021               *
1022               * Call to STLXP2 does next instruction
1023               *        D1=C
1024 02FCC 11B           C=R3
1025 02FCF CE            C=C-1  A
1026 02FD1 CE            C=C-1  A
1027               *
1028               * Set Restart Address; Position in Output line
1029               * Request Characterization nibble on return
1030               * RESCAN for "begin" token
1031               *
1032 02FD3 850    RESTA5 ST=1   0              Request characterization
1033 02FD6 841           ST=0   1
1034 02FD9 842           ST=0   2
1035 02FDC 843           ST=0   3
1036 02FDF 84B           ST=0   11
1037 02FE2 8E00          GOSUBL =RESCAN        Look for "begin" token
           00
1038               * Save RESTART address; Reset RESTART flag if non-mainframe token
1039 02FE8 8E00          GOSUBL =SVRST2        RESTORE D0 & SAVE INPUT PTR
           00
1040               *
1041 02FEE 868           ?ST=0  8              Not immediately after THEN/ELSE
1042 02FF1 60            GOYES  LINE#?
1043               * S-R0-0 only updated if S10 Set (INADDR positioned prior to tIF)
1044 02FF3 7000          GOSUB  =EXTIF+
1045               *
1046               * If (Line# found & Begin BASIC token) OR
1047               *    (Line# NOT found & Calc BASIC token)
1048               *      Save Restart Address
1049               *      Continue parsing
1050               *
1051 02FF7 865   =LINE#? ?ST=0  5              Line# NOT found ?
1052 02FFA 80            GOYES  RESTA7
1053 02FFC 8C00          GOLONG =LNEPLN        Continue parsing
           00
1054 03002 8C00   RESTA7 GOLONG =LNEP26        Calc BASIC keyword ?
           00
1055               *
1056 03008 133    RSTRT? AD1EX                 Save cursor position
1057 0300B 1F00          D1=(5) =S-R1-3
           000
1058 03012 1574          C=DAT1 S              Restart Flag
1059 03016 1CE           D1=D1- 15
1060 03019 147           C=DAT1 A              Read in original err#
1061 0301C A4E           C=C-1  S
1062 0301F 500           RTNNC                 Restart?
1063 03022 8AA           ?C=0   A              Stmt not earlier restarted?
1064 03025 B0            GOYES  RSTR?5
1065 03027 174           D1=D1+ 5
```

```
1066 0302A 143          A=DAT1 A            Original error position
1067 0302D 134          D0=C                Original error#
1068 03030 131   RSTR?5 D1=A                Cursor position
1069 03033 02           RTNSC
1070             *
1071 03035 1F00 =REST*  D1=(5) =S-R1-3
           000
1072 0303C 1574         C=DAT1 S
1073 03040 A4E          C=C-1  S
1074 03043 664F         GOTO   RESTxx
1075             *
```

```
1076                    STITLE Restore D1,C from R3 (D1C=R3)
1077                ******************************************************************
1078                ******************************************************************
1079                **
1080                ** Name:(S) D1C=R3  -  Restore C(A),D1 from R3
1081                **
1082                ** Category:   GENUTL
1083                **
1084                ** Purpose:
1085                **      Restores D1 from R3(5-9)
1086                **      Reverse effect of R3=D1C
1087                **
1088                ** Entry:
1089                **      None
1090                **
1091                ** Exit:
1092                **      C(A) = R3(A)
1093                **      A(A) = R3(5-9)
1094                **      D1   = R3(5-9)
1095                **      Carry preserved from entry
1096                **
1097                ** Calls:      None
1098                **
1099                ** Uses.......
1100                **  Exclusive: A,C(A),D1
1101                **  Inclusive: A,C(A),D1
1102                **
1103                ** Stk lvls:   0
1104                **
1105                ** History:
1106                **
1107                **    Date      Programmer              Modification
1108                **  --------    ----------   ----------------------------------
1109                **  07/07/82    JP           Modified documentation
1110                **
1111                ******************************************************************
1112                ******************************************************************
1113                *
1114                *
1115 03047 113     =D1C=R3 A=R3
1116 0304A D6               C=A    A
1117 0304C BF4              ASR    W
1118 0304F BF4              ASR    W
1119 03052 BF4              ASR    W
1120 03055 BF4              ASR    W
1121 03058 BF4              ASR    W
1122 0305B 131              D1=A
1123 0305E 01               RTN
```

```
1124                    STITLE Get Next Non-Blank Character
1125            ********************************************************************
1126            ********************************************************************
1127            **
1128            ** Name:(S) GNXTCR  -  Get Next Non-blank Character
1129            ** Name:(S) OAGNXT  -  Output byte, Get Next Non-blank Character
1130            ** Name:    GNXCR+  -  Get Next Non-blank Character
1131            **
1132            ** Category:   PARUTL
1133            **
1134            ** Purpose:
1135            **      Gets next non-blank character.
1136            **
1137            **      OAGNXT first outputs a byte from A(B) before scanning
1138            **      for the next non-blank character.
1139            **
1140            **      GNXCR+ first increments D1 by 2 before scanning for
1141            **      the next non-blank character.
1142            **
1143            ** Entry:
1144            **      OAGNXT - A(B) contains byte to output
1145            **               D(A) = (AVMEME)
1146            **               D0 points to where byte to be written
1147            **               D1 points to where to begin scanning for
1148            **               next non-blank character.
1149            **      GNXTCR - D1 points to where to begin scanning for
1150            **               next non-blank character.
1151            **      GNXCR+ - D1 points 2 nibbles prior to where to begin
1152            **               scanning for the next non-blank character.
1153            **
1154            ** Exit:
1155            **      D1 points to next non-blank character
1156            **      A(B)   =  Next non-blank Character
1157            **      C(B)   =  Ascii Blank
1158            **      P      =  0
1159            **      Carry set
1160            **
1161            **      If not enough memory to output byte, generates
1162            **      MEMERR (OAGNXT entry only)
1163            **
1164            ** Calls:      OUT1TK - (ONGNXT Only)
1165            **
1166            **  Uses:      A(B),C(B), D0 (ONGNXT Only),D1, P
1167            **
1168            ** Stk lvls:
1169            **   GNXTCR:   0
1170            **   GNXCR+:   0
1171            **   ONGNXT:   2
1172            **
1173            ** History:
1174            **
1175            **    Date     Programmer     Modification
1176            **  --------   ----------   ---------------
1177            **  07/07/82   JP           Modified Documentation
1178            **  09/24/82   FH           Modified Documentation
```

```
1179            **  11/02/83   S.W.         Fixed documentation header
1180            **
1181            ********************************************************************
1182            ********************************************************************
1183            *
1184 03060 778C =DAGNXT GOSUB  OUT1TK
1185 03064 1C1  =GNXTCR D1=D1- 2
1186 03067 20   =GNXCR+ P=     0
1187 03069 3102         LCASC  \ \
1188 0306D 171   GNXTC1 D1=D1+ 2
1189 03070 14B          A=DAT1 B
1190 03073 962          ?A=C   B
1191 03076 7F           GOYES  GNXTC1
1192 03078 02           RTNSC
1193            *
1194 0307A 06   =CRGJMP RSTK=C                Put address in C on stack
1195 0307C 01           RTN                   Jump to it
1196            *
```

```
1197                    STITLE DEF Parse
1198                ********************************************************************
1199                ********************************************************************
1200                **
1201                ** Name:    DEFP    -  DEF FN and DEF KEY Statement Parse
1202                **
1203                ** Category:   STPARS
1204                **
1205                ** Purpose:    Parses DEF FN and DEF KEY statements
1206                **
1207                ** Entry:      D1 past DEF keyword
1208                **             D0 past tDEF in output buffer
1209                **             D(A) = (AVMEME)
1210                **
1211                ** Exit:       If next token is tKEY
1212                **               then exit via DFKEYP
1213                **               else Return with carry clear
1214                **                    D1 past valid DEF FN statement
1215                **                    D0 past tokenized statement
1216                **
1217                ** Calls:      ORGNXT, OUTBYT, WRDSCN, IFCK, OUTNB4, USRFP,
1218                **             PRENCK, SPLVRP, RS-R03, OUTNIB
1219                **
1220                ** Uses:       A-C, S0-S3, S11, S7, S8, R0-R3
1221                **
1222                ** Stk Lvls: 4
1223                **
1224                ** Detail: TOKENIZED AS:
1225                ** tDEF<5 nib field>tFN tVAR [tSEMIC tCONSTANT tSEMIC]
1226                **  <1 nib parm ct.> [tPRMST <parm list>a)] [tCOMMA expression]
1227                **
1228                **      tSEMIC tCONSTANT connotes a string length declaration.
1229                **         tDEF is characterized as 'LEGAL AFTER THEN' & CALC
1230                **         BASIC.  This allows 'DEF KEY' to be executable.
1231                **         However 'DEF FN' isn't 'LEGAL AFTER THEN' or
1232                **         CALC BASIC.
1233                **
1234                **      tFN signifies a single line function.  In the case of a
1235                **         multiple line user defined function, this field will
1236                **         be a zero byte.
1237                **
1238                **      Because of the tokenization of the DEF FN statement,
1239                **         statement terminators can't have a low nibble of F.
1240                **
1241                ** History:
1242                **
1243                **    Date      Programmer   Modifications
1244                **  ---------  ----------   -------------
1245                **  02/01/83   S.W.         Updated documentation
1246                **
1247                ********************************************************************
1248                ********************************************************************
1249                *
1250                *
1251 0307E 6CAD  DEFPE+ GOTO   ERR01
```

```
1252                *
1253 03082 7000 =IFCK   GOSUB  =RS-R03
1254 03086 94A          ?C=0   S              Not in IF construct?
1255 03089 00           RTNYES
1256 0308B 64ED  DEFPE6 GOTO   ERR06          NON-EXECUTABLE
1257                *
1258 0308F 61FD  DFPEXP GOTO   ERR08          EXCESS CHARS
1259                *
1260 03093 739B =DEFP   GOSUB  WRDSCN
1261 03097 00           CON(2) =tKEY
1262 03099 000          REL(3) =DFKEYP
1263 0309C 00           CON(2) 0
1264                *
1265 0309E 3100         LC(2)  =tFN
1266 030A2 966          ?A#C   B              NOT T_FN?
1267 030A5 9D           GOYES  DEFPE+
1268                * DEF FN - DON'T WANT tFN OUTPUT
1269 030A7 77DF         GOSUB  IFCK           Read S-R0-3
1270 030AB 865          ?ST=0  5              CALC BASIC?
1271 030AE DD           GOYES  DEFPE6
1272                * ALLOW ROOM FOR 5 NIBBLE LINKING FIELD
1273 030B0 7000         GOSUB  =OUTNB4        Call OUTNB5
1274                *
1275                * tFN IS STILL IN A
1276                * SAVE POINTER TO WHERE tFN WILL GO
1277                *
1278 030B4 136          CDOEX
1279 030B7 134          DO=C
1280 030BA 10B          R3=C                  R3 WILL BE PRESERVED AS LONG AS NEC
1281 030BD 857          ST=1   7              IN DEF FN...
1282 030C0 76D0         GOSUB  USRFP
1283 030C4 D8           B=A    A              SAVE NEXT CHAR - A(B)
1284 030C6 132          ADOEX                 SAVE PTR TO WHERE PARM COUNT WILL G
1285 030C9 130          DO=A
1286 030CC 785C         GOSUB  OUTNIB         INCREMENT DO OVER PARM COUNT NIB
1287 030D0 BF0          ASL    W              A(0)=0; Save address in A(5-1)
1288 030D3 E4           A=A+1  A              INITIALIZE PARM COUNT (1)
1289 030D5 3182         LCASC  \(\
1290 030D9 965          ?B#C   B              NOT T_LPAREN?
1291 030DC F2           GOYES  DEFP70         IF NO, THEN NO PARM LIST
1292 030DE 102          R2=A                  SAVE AWAY PARM COUNT & PTR
1293 030E1 3100         LC(2)  =tPRMST
1294 030E5 7CFB         GOSUB  OUTBY+
1295 030E9 7E12  DEFP50 GOSUB  SPLVRP         Simple Variable Parse
1296                *
1297 030ED 11A          C=R2
1298 030F0 B06          C=C+1  P              INCR PARM COUNT
1299 030F3 4B9          GOC    DFPEXP         EXCESS PARMS?
1300 030F6 10A          R2=C
1301                *
1302 030F9 7000         GOSUB  =COMCK
1303 030FD 4BE          GOC    DEFP50         T_COMMA?
1304 03100 7000         GOSUB  =PRENCK
1305 03104 785F         GOSUB  ORGNXT
1306 03108 112          A=R2                  GET PARM COUNT (+1)
```

```
1307            *
1308 0310B CC    DEFP70 A=A-1  A             PARM COUNT - Decrement A(0)
1309 0310D D6           C=A    A             Read in A(0)
1310 0310F BF4          ASR    W             A(A) contains address
1311 03112 132          ADOEX                SAVE DO IN A
1312 03115 1540         DATO=C P             WRITE OUT PARM COUNT
1313 03119 130          DO=A                 RESTORE DO
1314 0311C 14B          A=DAT1 B             RE-READ CHARACTER
1315            *
1316 0311F 31D3         LCASC  \=\
1317 03123 962          ?A=C   B
1318 03126 13           GOYES  ASNMP1
1319            * A MULTIPLE-LINE FUNCTION - WRITE OO OVER tFN
1320 03128 113          A=R3
1321 0312B 132          ADOEX                SAVE DO IN A
1322 0312E D2           C=0    A             Zero out C(B)
1323 03130 14C          DATO=C B
1324 03133 130          DO=A                 RESTORE DO
1325 03136 03           RTNCC
1326            *
1327            *
1328            *
```

```
1329                    $TITLE FN Parse
1330            ********************************************************************
1331            ********************************************************************
1332            **
1333            ** Name:    FNP     -  Parse User-defined Function Assignment
1334            ** Name:    ASNMP   -  Assignment Parse
1335            **
1336            ** Category:   STPARS
1337            **
1338            ** Purpose:
1339            **              FNP entry is used to parse user-defined function
1340            **              assignments.
1341            **
1342            **              ASNMP entry parses the 'right side' of an
1343            **              assignment.
1344            **
1345            ** Entry:
1346            **      D(A) = (AVMEME)
1347            **      FNP - D0 points past statement length byte
1348            **            NTOKEN has already been called and tFN found.
1349            **            S10=0 => FN was preceded by LET
1350            **            S10=1 => Implied Let of FN
1351            **                     If S9=0
1352            **                       then
1353            **                         If this fails, try implied DISP.
1354            **                       else
1355            **                         Immediately after THEN/ELSE, so
1356            **                         If this fails, parse as label
1357            **
1358            **    ASNMP - D1 points at 'alleged' ascii =
1359            **            A(B) contains character D1 points to
1360            **            P=0
1361            **            Left side of = has already been parsed
1362            **            S8=1 iff left side was numeric
1363            **            D0 points past tokenization of left side
1364            **
1365            ** Exit:     via RESPTR, or error exit
1366            **
1367            ** Calls:    OUT1TK, OUTNIB, USRFP, EXPPR+, NTOKEN, OUTBYT
1368            **
1369            ** Uses:     S7, S8, R2, R0, R1, S0-S3, S11, FUNCD0, D(15-5)
1370            **
1371            ** Stk Lvls: 4
1372            **
1373            ** Detail:   TOKENIZED AS:
1374            **           tFN tVAR F t= expression
1375            **
1376            ** History:
1377            **
1378            **    Date     Programmer   Modifications
1379            **  --------   ----------   -------------
1380            **  03/06/83   S.W.         Documented
1381            **  07/01/83   S.W.         Don't error on FN assignment
1382            **                            if from Keyboard
1383            **
```

```
1384            ************************************************************************
1385            ************************************************************************
1386            *
1387 03138       =FNP
1388 03138 847   =FNP+    ST=0   7              IN FN...
1389 0313B 7000          GOSUB  =R3=D1*        RESPTR, Save D1 (in case of DISP)
1390 0313F 7000          GOSUB  =Ntoken        Re-tokenize FN<var>
1391 03143 7350          GOSUB  USRFP
1392 03147 30F           LCHEX  F
1393 0314A 7ADB          GOSUB  OUTNIB
1394            *
1395            * Evaluate right side of assignment
1396            *
1397 0314E 31D3 =ASNMP   LCASC  \=\
1398 03152 966           ?A#C   B
1399 03155 E3            GOYES  USRFE+         Syntax error
1400 03157 3100  ASNMP1  LC(2)  =tCOMMA
1401 0315B 768B          GOSUB  OUTBY+
1402 0315F 7711          GOSUB  EXPPR+
1403 03163 870           ?ST=1  0              INVALID?
1404 03166 62            GOYES  USRFPE
1405 03168 863           ?ST=0  3              STRING?
1406 0316B 61            GOYES  USRFP7
1407            * NUMERIC EXPR - CHECK FOR TYPE MATCH
1408 0316D 868           ?ST=0  8              STRING FUNCTION NAME?
1409 03170 61            GOYES  USFPE2
1410            *
1411            *
1412            *
1413            ************************************************************************
1414            ************************************************************************
1415            **
1416            ** Name:(S) RESPTR  -  Restore Input Pointer
1417            **
1418            ** Category:   PARUTL
1419            **
1420            ** Purpose:
1421            **      Restores D1 to its position prior to NTOKEN call
1422            **
1423            ** Entry:
1424            **      (LEXPTR) = address of input pointer (advanced past
1425            **                 leading blanks) prior to last call to
1426            **                 NTOKEN.
1427            **
1428            ** Exit:
1429            **      D1 re-positioned
1430            **      Carry clear
1431            **
1432            ** Calls:      none
1433            **
1434            ** Uses:       A(A), D1
1435            **
1436            ** Stk lvls:   0
1437            **
1438            ** History:
```

```
1439                **
1440                **    Date     Programmer             Modification
1441                **  --------   ----------   ----------------------------------
1442                **  07/08/82   S.W.         Added documentation
1443                **
1444                ********************************************************************
1445                ********************************************************************
1446 03172 1F00 =RESPTR D1=(5) =LEXPTR
           000
1447 03179 143           A=DAT1 A
1448 0317C 131           D1=A
1449 0317F 03            RTNCC
1450                *
1451                ********************************************************************
1452                ********************************************************************
1453                **
1454                ** Name:    USRFP   -  User-Defined Function Name Parse
1455                **
1456                ** Category:   PARUTL
1457                **
1458                ** Purpose:
1459                **      Parses user-defined function name
1460                **
1461                ** Entry:
1462                **      P      =  0
1463                **      D(A)   =  (AVMEME)
1464                **      S7=1 =>DEF FN parse (string length declaration valid)
1465                **             D0 past tDEF in output buffer
1466                **      S7=0 =>FN assignment (string length decl invalid)
1467                **             D0 past statement length byte
1468                **      A = Entire function name tokenization, as returned
1469                **          by NTOKEN.  ( A(B)=tFN )
1470                **      D1 points past function name (as per NTOKEN exit)
1471                **
1472                ** Exit:
1473                **
1474                **      Valid syntax =>
1475                **        Return with carry set
1476                **        P=0
1477                **        User-defined function name parsed & output
1478                **        D1 past user defined function name
1479                **        A(B) = 1st non-blank character to follow parsed
1480                **               input stream.
1481                **        D0 past tokenized user-defined function name.
1482                **        If S7=1 on entry,
1483                **           then any string length declaration was also
1484                **                parsed, with D1 past the optional
1485                **                declaration, and D0 past its tokenization.
1486                **
1487                **      Invalid syntax =>
1488                **        Error exit to PARERR
1489                **
1490                ** Calls:      OUTVAR, GNXTCR, OUTBY+, NTOKEN, CONCOM, OAGNXT
1491                **
1492                ** Uses.......
```

```
1493                ** Exclusive: A-C, R2, S8
1494                ** Inclusive: A-C, R0,R2, S0-S3,S8,S11
1495                **
1496                ** Stk lvls:   3
1497                **
1498                ** History:
1499                **
1500                **    Date     Programmer              Modification
1501                ** --------   ----------   ---------------------------------
1502                ** 03/06/83   S.W.         Routine documented
1503                **
1504                ********************************************************************
1505                ********************************************************************
1506                *
1507 03181 868     USRFP7 ?ST=0  8              STRING FUNCTION NAME?
1508 03184 EE              GOYES  RESPTR
1509                *
1510 03186 11A     USFPE2 C=R2
1511 03189 135             D1=C                 RESTORE PTR
1512                *
1513 0318C 854     USRFPE ST=1   4
1514 0318F 65AC            GOTO   ERR02         Invalid Expr
1515                *
1516 03193 854     USRFE+ ST=1   4
1517 03196 649C    USRFE1 GOTO   ERR01
1518                *
1519 0319A D8      USRFP  B=A    A
1520 0319C 848            ST=0   8
1521 0319F F5             BSR    A
1522 031A1 BB5            BSR    X
1523 031A4 3100           LC(2)  =tSVAR
1524 031A8 961            ?B=C   B
1525 031AB 50             GOYES  USRFP2
1526 031AD 858            ST=1   8              NUMERIC FLAG
1527 031B0 7000    USRFP2 GOSUB  =OUTVAR
1528 031B4 7CAE           GOSUB  GNXTCR
1529 031B8 867            ?ST=0  7              NOT IN DEF FN...?
1530 031BB 00             RTNYES
1531                *
1532 031BD 31B5           LCASC  \[\
1533 031C1 966            ?A#C   B              NO STRING LEN DECL?
1534 031C4 00             RTNYES
1535                *
1536 031C6 878            ?ST=1  8              NUMERIC?
1537 031C9 DC             GOYES  USRFE1         SYNTAX ERROR
1538                *
1539 031CB 3100           LC(2)  =tSEMIC
1540 031CF 721B           GOSUB  OUTBY+         STEP OFF '['
1541                * MUST BE CONSTANT INSIDE BRACKETS
1542 031D3 7000           GOSUB  =Ntoken
1543 031D7 8E00           GOSUBL =CONCOM
           00
1544 031DD 48B            GOC    USRFE1         NOT A CONSTANT?
1545 031E0 3100           LC(2)  =tSEMIC        NEEDED FOR CONSTANT DECOMPILE
1546 031E4 DA             A=C    A
```

```
1547 031E6 767E          GOSUB  OAGNXT          OUTPUT TOKEN, ETC.
1548 031EA 31D5          LCASC  \]\
1549 031EE 966           ?A#C   B
1550 031F1 2A            GOYES  USRFE+          NO CLOSING BRACKET?
1551             *
1552 031F3 637E          GOTO   GNXCR+
1553             *
1554             *
```

```
1555                    STITLE READ/INPUT/LINPUT Parse
1556            ********************************************************************
1557            ********************************************************************
1558            **
1559            ** Name:     READP   -  READ, READ# Statement Parse
1560            ** Name:     INPUTP  -  INPUT Statement Parse
1561            ** Name:     LINPTP  -  LINPUT Statement Parse
1562            ** Name:     DSTp    -  Single Destination Variable Parse
1563            ** Name:(S)  READP5  -  Destination Variable List Parse
1564            **
1565            ** Category: STPARS
1566            **
1567            ** Purpose:  Parses READ, READ#, INPUT, LINPUT statements.
1568            **
1569            **           DSTp entry expects a 'destination' variable, ie
1570            **           one that is suitable for storing a value.
1571            **
1572            **           READP5 entry will parse a list of destination
1573            **           variables, delimited by commas.  Depending on
1574            **           status bits S8 and S9 on entry, it allows or
1575            **           disallows dummy arrays, allows a list of any
1576            **           number of destination variables, or demands
1577            **           that the first variable in the list is a
1578            **           string destination and then returns to leave
1579            **           the rest of the parse (if any) to the caller.
1580            **
1581            ** Entry:    D(A) = (AVMEME)
1582            **           5 entry points:
1583            **           1) LINPTP - D1 past LINPUT
1584            **                       D0 past tLINPT
1585            **           2) INPUTP - S9=0
1586            **                       D1 past INPUT
1587            **                       D0 past tINPUT
1588            **           3) READP  - S8=0, S9=0
1589            **                       D1 past READ
1590            **                       D0 past tREAD
1591            **           4) READP5 - S8=0 iff Dummy arrays are valid
1592            **                       S9=1 iff single string var parse
1593            **           5) DSTp   - D1 pts to alleged destination var.
1594            **
1595            ** Exit:     Valid parse =>
1596            **             P=0
1597            **             LINPTP,INPUTP,READP entry:
1598            **              D1 past syntactically correct stmt
1599            **              D0 past tokenized statement
1600            **              Return with carry clear
1601            **
1602            **             READP5 entry:
1603            **              D1 past the parsed variable or var list
1604            **              D0 past tokenized destination variable(s)
1605            **              Return with carry clear
1606            **              If S9=1 on entry
1607            **               Single string destination variable parsed
1608            **               D1 past the string variable
1609            **               D0 past the tokenized string variable
```

```
1610            **
1611            **              DSTp entry:
1612            **               D1 past destination variable
1613            **               D0 past tokenized destination variable
1614            **               Carry set on return iff dummy array
1615            **
1616            **            Invalid parse =>
1617            **              LINPTP,INPUTP,READP entry:
1618            **               Error exit to PARERR
1619            **
1620            **              READP entry:
1621            **               Error exit to PARERR
1622            **               If S8=0,S9=0 on entry
1623            **                Something in list was not a destination
1624            **                variable, or a delimiter was missing
1625            **               If S8=1,S9=0 on entry
1626            **                Something in list was either not a
1627            **                destination variable, or was a dummy
1628            **                array, or a delimiter was missing
1629            **               If S8=0,S9=1
1630            **                First item in list was not a string
1631            **                destination variable.
1632            **               If S8=1,S9=1
1633            **                First item in list was either a dummy
1634            **                array or was not a string destination
1635            **                variable.
1636            **
1637            **              DSTp entry:
1638            **                Input either was an invalid expression
1639            **                or was inappropriate as a destination.
1640            **
1641            ** Calls:     OUT1TK, NTOKEN, DSTp, COMCK, PILP, WRDSCN
1642            **            DATACK, STRGCK, COMCK1, OUT1TK, EXPPR+
1643            **
1644            ** Uses:      A-C,D(15-5), D1,D0, R0-R2, S0-S3,S7-S9,S11
1645            **            FUNCD0, P
1646            **
1647            ** Detail:    Doesn't allow for INPUT/READ/LINPUT without at
1648            **            least one variable in the list
1649            **            Allows for READ#, but not INPUT#.
1650            **            READ# compiled as:
1651            **            # num expr [tCOMMA <num expr>] [SEMIC <var list>]
1652            **              Even if there's no record# specified, there must
1653            **              be a variable list.
1654            **
1655            **            INPUTP and LINPTP allow an optional prompt
1656            **            and initial string for default values
1657            **
1658            **            Tokenized destination variables in READ, READ#,
1659            **            INPUT and LINPUT are delimited by tCOMMA.
1660            **
1661            ** Stk lvls: 5
1662            **
1663            ** History:
1664            **
```

```
1665                ** Date     Programmer   Modifications
1666                ** --------  ----------   --------------
1667                ** 12/06/82  S.W.         READ,READ# allows dummy arrays
1668                ** 03/11/83  S.W.         Tokenize INPUT with prompt with
1669                **                         preceding zero byte
1670                ** 05/18/83  S.W.         Calls new subroutine: DSTp
1671                **
1672                ************************************************************
1673                ************************************************************
1674                *
1675 031F7 859   =LINPTP ST=1   9
1676 031FA D0    =INPUTP A=0    A              Output zero byte in case of prompt
1677 031FC 7BEA          GOSUB  OUT1TK
1678 03200 7000          GOSUB  =DATACK
1679 03204 858           ST=1   8              INPUT flag - no dummy arrays
1680 03207 580           GONC   INPTP1         Found input prompt?
1681 0320A 181           D0=D0- 2              Back up over zero byte token
1682 0320D 5D2           GONC   READP5         (B.E.T.)
1683
1684                *
1685 03210 761A  INPTP1 GOSUB  WRDSCN
1686 03214 00            CON(2) =tSEMIC
1687 03216 520           REL(3) READP5
1688 03219 00            CON(2) =tCOMMA
1689 0321B 800           REL(3) INPTP2
1690 0321E 00            CON(2) 0
1691                *
1692 03220 521           GONC   INPTER         (B.E.T.)
1693                *
1694 03223 181   INPTP2 D0=D0- 2              Back over tokenized comma.
1695 03226 7000          GOSUB  =STRGCK
1696 0322A 3100          LC(2)  =tSEMIC
1697 0322E 962           ?A=C   B
1698 03231 22            GOYES  READP4
1699                *
1700 03233 67FB  INPTER GOTO   ERR01          SYNTAX ERROR
1701                *
1702 03237 7000 =READP  GOSUB  =PILP
1703                * S8=1 => Dummy array not legal
1704                * S9=1 => LINPUT parse
1705 0323B 7B10 =READP5 GOSUB  DSTp
1706 0323F 570           GONC   READP2         Not dummy array ?
1707 03242 878           ?ST=1  8              INPUT parse?
1708 03245 62            GOYES  REDPE5
1709 03247 879   READP2 ?ST=1  9              LINPUT parse?
1710 0324A D3            GOYES  LNPTP5
1711 0324C 7000          GOSUB  =COMCK1
1712 03250 555           GONC   DECP01
1713 03253 749A  READP4 GOSUB  OUT1TK          OUTPUT COMMA
1714 03257 53E           GONC   READP5         (B.E.T.)
1715                *
1716                * Destination Parse - uses 4 stack levels
1717                *
1718 0325A 7C10 =DSTp   GOSUB  EXPPR+
1719 0325E 400           RTNC                  Dummy Array?
```

```
1720                * MUST CHECK FOR LEGAL EXPR AND DESTINATION
1721 03261 870           ?ST=1  0              INVALID EXPR?
1722 03264 70            GOYES  REDPE5
1723 03266 831           ?XM=0                 LEGAL DEST?
1724 03269 F0            GOYES  rtncc
1725                *
1726 0326B 11A  =REDPE5  C=R2                  RESTORE INPUT PTR
1727 0326E 135           D1=C
1728 03271 854           ST=1   4              NO RESTORE OF INPUT PTR
1729 03274 61FB  DECPE5  GOTO   ERR05          ILLEGAL VARIABLE
1730                *
1731 03278 03     rtncc  RTNCC
1732                *
1733 0327A 137    EXPPR+ CD1EX
1734 0327D 135           D1=C
1735 03280 10A           R2=C                  Save D1 in R2
1736 03283 6000          GOTO   =exppar
1737                *
1738 03287 863    LNPTP5 ?ST=0  3              String variable?
1739 0328A C1            GOYES  DECP01         RESPTR
1740 0328C 5ED           GONC   REDPE5         (B.E.T.)
1741                *
```

```
1742                    STITLE Declaration Statements Parse
1743                ********************************************************************
1744                ********************************************************************
1745                **
1746                ** Name:(S) DECP    -  Parse of Variable Declaration Statements
1747                **
1748                ** Category:   STPARS
1749                **
1750                ** Purpose:    Parses REAL, SHORT, INTEGER statements
1751                **
1752                ** Entry:      D1 past REAL, SHORT, or INTEGER keywords
1753                **             D0 past tREAL, tSHORT, or tINTEG
1754                **             D(A) = (AVMEME)
1755                **
1756                ** Exit:       If valid statement syntax:
1757                **              via RESPTR (Carry clear)
1758                **              D1 past syntactically correct statement
1759                **              D0 past tokenized statement in output buffer
1760                **
1761                **             If error in syntax:
1762                **              Exit to PARERR
1763                **
1764                ** Calls:    COMCK0, ARRYCK, VARP
1765                **
1766                ** Uses:     A-C,D(15-5), D0,D1, S0-S3,S11, R0,R1, FUNCD0
1767                **
1768                ** Stk lvls: 6
1769                **
1770                ** History:
1771                **
1772                **     Date    Programmer   Modifications
1773                **  ---------  ----------   -------------
1774                **  03/06/83   S.W.         New documentation header added
1775                **  05/10/83   S.W.         Added call to COMCK0
1776                **
1777                ********************************************************************
1778                ********************************************************************
1779                *
1780 0328F 7000 =DECP   GOSUB  =VARP
1781 03293 50E          GONC   DECPE5          Not a numeric variable ?
1782 03296 862          ?ST=0  2
1783 03299 60           GOYES  DECP00          Simple variable ?
1784 0329B 7000         GOSUB  =ARRYCK
1785 0329F 7700  DECP00 GOSUB  COMCK0
1786 032A3 4BE          GOC    DECP            Comma found & output ?
1787                *
1788 032A6 6BCE  DECP01 GOTO   RESPTR
1789                *
1790                *
1791                ********************************************************************
1792                ********************************************************************
1793                **
1794                ** Name:    COMCK0  -  Check Comma & Output Comma Token
1795                ** Name:(S) COMCK+  -  Check Comma & Output Comma Token
1796                **
```

```
1797            ** Category:   PARUTL
1798            **
1799            ** Purpose:
1800            **      Checks for tCOMMA & outputs it if found.
1801            **
1802            **      COMCK0 entry requires that NTOKEN be called
1803            **      before checking for tCOMMA.
1804            **
1805            **      COMCK+ entry assumes that NTOKEN has already
1806            **      been called.
1807            **
1808            ** Entry:
1809            **      D(A)    =  (AVMEME)
1810            **      D0      =  pointer to where tCOMMA to be output
1811            **      2 entry points:
1812            **      1) COMCK0 - D1 at opt. preceding blanks before
1813            **                  alleged comma.
1814            **      2) COMCK+ - A(B) contains byte to compare against
1815            **                  tCOMMA.
1816            **
1817            ** Exit:
1818            **      P       =  0
1819            **      Carry set => tCOMMA found & output
1820            **                   D0 incremented past tCOMMA
1821            **                   COMCK0 entry:
1822            **                    D1 pts past ascii comma
1823            **                   COMCK+ entry:
1824            **                    D1 preserved from entry
1825            **
1826            **      Carry clr => tCOMMA NOT found
1827            **                   D0 preserved from entry
1828            **                   COMCK0 entry:
1829            **                    A(B) = token found
1830            **                    D1 advanced past corresponding text
1831            **                   COMCK+ entry:
1832            **                    A(B) preserved from entry
1833            **                    D1 preserved from entry
1834            **
1835            **      If tCOMMA found, but not enough memory to
1836            **      output it, exits to MEMERR
1837            **
1838            ** Calls:       NTOKEN, COMCK1
1839            **
1840            ** Uses:        C, D0, P (COMCK+ entry)
1841            **              A-C, D1,D0, S0-S3,S11, R0, P (COMCK0 entry)
1842            **
1843            ** Stk lvls:    3
1844            **
1845            ** History:
1846            **
1847            **    Date      Programmer              Modification
1848            **  --------    ----------    ----------------------------------
1849            **  05/11/83    S.W.          Added documentation
1850            **
1851            ****************************************************************
```

```
1852            ****************************************************************
1853            * Carry set => comma found & output
1854            *
1855 032AA 7000 =COMCK0 GOSUB  =Ntoken
1856 032AE 7000 =COMCK+ GOSUB  =COMCK1
1857 032B2 500          RTNNC                  Not a comma ?
1858 032B5 723A         GOSUB  OUT1TK
1859 032B9 02           RTNSC
1860            *
```

```
1861                    STITLE DIM Parse
1862                ********************************************************************
1863                ********************************************************************
1864                **
1865                ** Name:    DIMP    -  DIM Statement Parse
1866                **
1867                ** Category:   STPARS
1868                **
1869                ** Purpose:    Parses DIM statement
1870                **
1871                ** Entry:      D(A) = (AVMEME)
1872                **             D1 past DIM
1873                **             D0 past tDIM in output buffer
1874                **
1875                ** Exit:       If valid statement parse:
1876                **              Carry clear on return
1877                **              D1 past syntactically correct statement
1878                **              Tokenized statement in output buffer
1879                **              D0 past tokenized statement
1880                **              P=0
1881                **             else
1882                **              Error exit through PARERR
1883                **
1884                ** Calls:    OUT1TK, VARP, ARRYCK, COMCK, NUMCK
1885                **
1886                ** Uses:     A-C,D(15-5),D0,D1,S0-S3,S7,S8,S11,R0,R1,FUNCD0
1887                **
1888                ** Detail:   Only allows 1 dimension for string array
1889                **
1890                ** Stack lvl: 5
1891                **
1892                ** History:
1893                **
1894                **    Date     Programmer   Modifications
1895                **  --------  ----------   ------------
1896                **  11/20/82   S.W.         DIM now allows simple variables
1897                **  05/11/83   S.W.         Added call to COMCK0; deleted
1898                **                           corresponding call to OUT1TK
1899                **  06/22/83   S.W.         Added clearing of S8 (SR#955-5)
1900                **
1901                ********************************************************************
1902                ********************************************************************
1903                *
1904                *
1905 032BB 848  =DIMP   ST=0   8              Needed after parse of strg w/o len
1906 032BE 7000        GOSUB  =VARP
1907 032C2 450         GOC    DIMP05         NUMERIC VARIABLE?
1908                * STRING VARIABLE
1909 032C5 858         ST=1   8              STRING FLAG
1910 032C8 862   DIMP05 ?ST=0  2              SIMPLE VARIABLE?
1911 032CB 21          GOYES  DIMP15
1912                * ARRAY VARIABLE
1913 032CD 7000        GOSUB  =ARRYCK
1914                *
1915 032D1 868         ?ST=0  8              NUMERIC ARRAY?
```

```
1916 032D4 90           GOYES  DIMP15
1917 032D6 CD           B=B-1  A              Decrement B(0)
1918 032D8 90D          ?B#0   P              More than 1 SUBSCRIPT?
1919 032DB 77           GOYES  TRCPE1
1920              *
1921 032DD 79CF  DIMP15 GOSUB  COMCKO
1922 032E1 49D          GOC    DIMP           COMMA?
1923              * NOT A COMMA - "[" OR EOL
1924 032E4 868          ?ST=0  8              NUMERIC? => "[" NOT LEGAL
1925 032E7 FB           GOYES  DECP01
1926              * STRING VARIABLE - CHECK FOR "["
1927 032E9 31B5         LCASC  \[\
1928 032ED 966          ?A#C   B
1929 032F0 6B           GOYES  DECP01
1930              * FOUND BRACKET
1931 032F2 3100  DIMP20 LC(2)  =tSEMIC
1932 032F6 DA           A=C    A
1933 032F8 7000         GOSUB  =NUMCKO        PARSE STRING LENGTH
1934 032FC 31D5         LCASC  \]\
1935 03300 966          ?A#C   B
1936 03303 F4           GOYES  TRCPE1
1937 03305 848          ST=0   8
1938 03308 54D          GONC   DIMP15         (B.E.T.)
1939              *
1940              ********************************************************************
1941              ********************************************************************
1942              **
1943              ** Name:    SPLVRP  -  Simple Variable Parse
1944              **
1945              ** Category:   PARUTL
1946              **
1947              ** Purpose:
1948              **      Parses a simple variable
1949              **
1950              ** Entry:
1951              **      D1 at supposed simple variable
1952              **      D0 points to where tokenized variable to be written
1953              **      D(A) = (AVMEME)
1954              **
1955              ** Exit:
1956              **      P      =  0
1957              **      Return to caller =>
1958              **        Simple variable found & tokenization written out
1959              **        Carry set
1960              **      Else error exit (eIVVAR or MEMERR)
1961              **
1962              ** Calls:      VARP
1963              **
1964              ** Uses.......
1965              **      A-C, D0,D1, S0-S3,S11, R0
1966              **
1967              ** Stk lvls:   4
1968              **
1969              ** History:
1970              **
```

```
1971            **    Date     Programmer             Modification
1972            **  --------   ----------   ----------------------------------
1973            **  02/03/83   S.W.         Added documentation
1974            **
1975            ************************************************************
1976            ************************************************************
1977            *
1978 0330B 7000  SPLVRP GOSUB  =VARP          Error if not variable
1979 0330F 862          ?ST=0  2              Not an array variable?
1980 03312 00           RTNYES
1981            *
1982 03314 615B  DIMPE5 GOTO   ERR05          ILLEGAL VARIABLE
```

```
1983                    STITLE DESTROY/TRACE Parse
1984                ********************************************************************
1985                ********************************************************************
1986                **
1987                ** Name:     DSTRYP  -  DESTROY statement parse
1988                ** Name:     TRACEP  -  TRACE statement parse
1989                **
1990                ** Category:   STPARS
1991                **
1992                ** Purpose:    Parses DESTROY & TRACE statements
1993                **
1994                ** Entry:      D(A) = (AVMEME)
1995                **             DO pts past into output buffer
1996                **              (past tTRACE or tDSTRY)
1997                **             2 entry points:
1998                **             1) DSTRYP - D1 points past DESTROY
1999                **             2) TRACEP - D1 points past TRACE
2000                **
2001                ** Exit:       If valid statement parse:
2002                **              Return with carry clear
2003                **              P=0
2004                **              D1 past syntactically correct statement
2005                **              Tokenized statement written to output buffer
2006                **              DO advanced past end of tokenized statement
2007                **
2008                **             Else
2009                **              Error exit
2010                **
2011                ** Calls:    NTOKEN, RESPTR, COMCKO, COMCK, GNXTCR, WRDSCN
2012                **           SPLVRP
2013                **
2014                ** Uses:     A-C, R0-R2, S0-S3,S11
2015                **
2016                ** Stk lvls: 4
2017                **
2018                ** Detail:   DESTROY [ALL|simple variable list]
2019                **             TRACE [FLOW|OFF|VARS]
2020                **
2021                ** History:
2022                **
2023                **    Date     Programmer   Modifications
2024                **  --------   ----------   -------------
2025                **  03/06/83   S.W.         New documentation header added
2026                **
2027                ********************************************************************
2028                ********************************************************************
2029                *
2030 03318 7E09 =DSTRYP GOSUB  =WRDSCN
2031 0331C 00           CON(2) =tALL
2032 0331E E2C          REL(3) lnep66
2033 03321 00           CON(2) 0
2034 03323 7B4E         GOSUB  RESPTR
2035                *
2036 03327 70EF  DSTRPO GOSUB  SPLVRP         Simple Variable Parse
2037 0332B 7B7F         GOSUB  COMCKO
```

```
2038 0332F 554          GONC   DSTRP5         NOT COMMA?
2039 03332 44F          GOC    DSTRP0         (B.E.T.)
2040              *
2041 03335 71F8 =TRACEP GOSUB  =WRDSCN
2042 03339 00           CON(2) =tOFF
2043 0333B 11C          REL(3) lnep66
2044 0333E FE10         CON(6) =tVARS
           B5
2045 03344 80C          REL(3) lnep66
2046 03347 FE10         CON(6) =tFLOW
           92
2047 0334D FFB          REL(3) lnep66
2048 03350 00           CON(2) 0
2049              *
2050 03352 68DA  TRCPE1 GOTO   ERR01
```

```
2051                    STITLE #CK
2052                ********************************************************************
2053                ********************************************************************
2054                **
2055                ** Name:(S) #CK     -  Check for #
2056                **
2057                ** Category:   PARUTL
2058                **
2059                ** Purpose:
2060                **      Compares next non-blank character against ascii #
2061                **
2062                ** Entry:
2063                **      D1 points at optional blanks preceding character to
2064                **       compare against
2065                **
2066                ** Exit:
2067                **      P      =  0
2068                **      D1 points to next non-blank character
2069                **      A(B)   =  Next non-blank character
2070                **      Carry clear => Character is #
2071                **      Carry set   => Character is not #
2072                **
2073                ** Calls:      GNXTCR
2074                **
2075                ** Uses:       A(B), C(B), D1, P
2076                **
2077                ** Stk lvls:   1
2078                **
2079                ** History:
2080                **
2081                **    Date     Programmer              Modification
2082                **  --------   ----------   ---------------------------------
2083                **  11/03/83   S.W.         Added documentation header
2084                **
2085                ********************************************************************
2086                ********************************************************************
2087                *
2088 03356 7A0D =#CK    GOSUB  GNXTCR
2089 0335A 3132         LCASC  \#\
2090 0335E 966          ?A#C   B
2091 03361 00           RTNYES
2092 03363 01           RTN
2093                *
2094                *
```

```
2095                STITLE OFF Parse
2096          ********************************************************************
2097          ********************************************************************
2098          **
2099          ** Name:    OFFP    -  OFF Statement Parse
2100          **
2101          ** Category:   STPARS
2102          **
2103          ** Purpose:
2104          **      Parses OFF Statement
2105          **
2106          ** Entry:
2107          **      D(A) = (AVMEME)
2108          **      D0 points into output buffer past tOFF
2109          **      D1 points past OFF
2110          **
2111          ** Exit:
2112          **      Valid statement syntax =>
2113          **       Return with carry clear (via RESPTR)
2114          **       P=0
2115          **       D1 points past syntactically correct stmt
2116          **       Tokenized statement written to output buffer
2117          **       D0 points past tokenized statement
2118          **
2119          **      Else error exit
2120          **
2121          ** Calls:      WRDSCN, #CK, NUMCK (golong)
2122          **
2123          ** Uses.......
2124          **  Exclusive: A,C,D0,D1
2125          **  Inclusive: A-C,D(15-5),D0,D1,R0,R1,S0-S3,S7,S11,FUNCD0
2126          **
2127          ** Stk lvls:   4
2128          **
2129          ** Detail:
2130          **      OFF [ ERROR | TIMER   # <timer no> ]
2131          **
2132          ** Algorithm:
2133          **
2134          **      If next token = ERROR  (WRDSCN outputs the token)
2135          **         Return
2136          **      If next token = TIMER  (WRDSCN outputs the token)
2137          **         If next char # "#"
2138          **            Error Exit
2139          **         Skip #
2140          **         Verify <timer no> expression (NUMCK)
2141          **         Restore pointer & Return  (through RESPTR)
2142          **      else
2143          **         Restore pointer & return
2144          **
2145          ** History:
2146          **
2147          **    Date     Programmer             Modification
2148          **  --------   ----------   ----------------------------------
2149          **  07/07/82   JP           Modified documentation
```

```
2150                **
2151                ********************************************************************
2152                ********************************************************************
2153 03365 71C8 =OFFP   GOSUB  WRDSCN
2154 03369 00           CON(2) =tERROR        OFF ERROR
2155 0336B 9F8          REL(3) WRDS35         Return CC
2156 0336E 00           CON(2) =tTIMER        OFF TIMER
2157 03370 900          REL(3) OFFP10
2158 03373 00           NIBHEX 00             OFF
2159 03375 6CFD  DSTRP5 GOTO   RESPTR         Restore ptr & return
2160                *
2161                * OFF TIMER
2162                *
2163 03379 79DF  OFFP10 GOSUB  #CK            Get next character
2164 0337D 4F3          GOC    FREE07         Not # ?
2165 03380 171          D1=D1+ 2              Skip #
2166 03383 8C00 =FIXp   GOLONG =FIXP          NUMCK, RESPTR
           00
2167                *
2168                *
```

```
2169                    STITLE FREE/CLAIM Parse
2170            ********************************************************************
2171            ********************************************************************
2172            **
2173            ** Name:     FREEp   -  FREE and CLAIM Statement Parse
2174            ** Name:     SHOWp   -  SHOW Statement Parse
2175            ** Name:     FREE15  -  Parse Numeric Expression, Right Paren
2176            **
2177            ** Category:   STPARS
2178            **
2179            ** Purpose:    Parse FREE, CLAIM and SHOW statements.
2180            **
2181            **             FREE15 entry parses a numeric expression and
2182            **             expects a closing parentheses
2183            **
2184            ** Entry:      D(A) = (AVMEME)
2185            **             D0 points into output buffer
2186            **             D1 points to ascii input stream
2187            **             3 entry points:
2188            **             1) SHOWp  - D1 past SHOW
2189            **                         D0 past tSHOW
2190            **             2) FREEp  - D1 past FREE or CLAIM
2191            **                         D0 past tFREE or tCLAIM
2192            **                         S8=0
2193            **             3) FREE15 - D1 pts at alleged numeric expr
2194            **
2195            ** Exit:       SHOWp & FREEp entry
2196            **             If valid statement parse:
2197            **              Return with carry clear
2198            **              P=0
2199            **              D1 past syntactically correct statement
2200            **              Tokenized statement written to output buffer
2201            **              D0 past tokenized statement
2202            **
2203            **             Else error exit
2204            **
2205            **             FREE15 entry
2206            **             Numeric expression followed by right paren:
2207            **              Carry clear
2208            **              P=0
2209            **              D1 past right parentheses
2210            **              Tokenized numeric expression written out
2211            **              D0 past tokenized numeric expression
2212            **
2213            **             Else error exit
2214            **
2215            ** Calls:    WRDSCN, PRENCK, GNXTCR, NUMCK, LPRNCK
2216            **
2217            ** Uses:     A-C,D(15-5),D0,D1,R0-R2,S0-S3,S7,S8,S11,FUNCD0
2218            **
2219            ** Stk lvls: 4
2220            **
2221            ** Detail:   FREE PORT (<num expr>)
2222            **           CLAIM PORT (<num expr>)
2223            **           SHOW PORT
```

```
2224            **
2225            ** History:
2226            **
2227            **    Date     Programmer   Modifications
2228            **  ---------  ----------   -------------
2229            **  02/01/83   S.W.         Updated documentation
2230            **  05/02/83   S.W.         Added call to LPRNCK
2231            **
2232            ********************************************************************
2233            ********************************************************************
2234            *
2235 03389 858  =SHOWp  ST=1   8              No port# allowed.
2236            *
2237 0338C      =CLAIMp
2238 0338C 74DC =FREEp  GOSUB  GNXTCR         FREE/CLAIM PORT
2239 03390 31A3         LCASC  \:\
2240 03394 966          ?A#C   B              Char after FREE/CLAIM = ":"?
2241 03397 50           GOYES  FREE03         No
2242 03399 171          D1=D1+ 2              Yes, skip it.
2243 0339C 7A88  FREE03 GOSUB  WRDSCN
2244 033A0 00           CON(2) =tPORT
2245 033A2 800          REL(3) FREE05
2246 033A5 00           NIBHEX 00             Other
2247 033A7 581          GONC   FREE08         (B.E.T.) Syntax error
2248            *
2249 033AA 878   FREE05 ?ST=1   8
2250 033AD E0           GOYES  RTNCC          SHOW parse ?
2251 033AF 7110         GOSUB  LPRNCK
2252            *
2253            * FSPECp entry
2254            *
2255 033B3 7000 =FREE15 GOSUB  =NUMCK         Parse numeric expression
2256 033B7 7000         GOSUB  =PRENCK
2257 033BB 03    RTNCC  RTNCC
2258            *
2259 033BD 854  =FREE07 ST=1   4
2260 033C0 6A6A  FREE08 GOTO   ERR01
2261            *
```

```
                        STITLE LPRNCK
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:    LPRNCK  -  Checks if Next Non-blank Char is (
                    **
                    ** Category:   PARUTL
                    **
                    ** Purpose:
                    **      Checks if next non-blank character is a left paren.
                    **      If it is not, LPRNCK takes an error exit.
                    **
                    ** Entry:
                    **      D1 points to opt. blanks preceding character to check
                    **
                    ** Exit:
                    **      If next non-blank character is a left parenthesis:
                    **       Return with carry clear
                    **       P=0
                    **       D1 advanced over the ascii (
                    **
                    **      Else error exit through PARERR
                    **
                    ** Calls:      GNXTCR
                    **
                    ** Uses:       A(B), C(B), D1, P
                    **
                    ** Stk lvls:   1
                    **
                    ** History:
                    **
                    **    Date     Programmer              Modification
                    **  --------   ----------   ----------------------------------
                    **  11/03/83   S.W.         Added documentation header
                    **
                    ****************************************************************
                    ****************************************************************
                    *
033C4 7C9C =LPRNCK  GOSUB  GNXTCR
033C8 3182          LCASC  \(\
033CC 966           ?A#C   B
033CF EE            GOYES  FREE07
033D1 171           D1=D1+ 2
033D4 01            RTN
                    *
```

```
=!CK     Abs   11788 #02E0C -   579
 !CK+    Abs   11804 #02E1C -   585
=!CK2    Abs   11795 #02E13 -   581
=!CK3    Abs   11807 #02E1F -   587
=#CK     Abs   13142 #03356 -  2088  2163
 ARRYCK  Ext                 -  1784  1913
=ASNMP   Abs   12622 #0314E -  1397
 ASNMP1  Abs   12631 #03157 -  1400  1318
=CLAIMp  Abs   13196 #0338C -  2237
 COMCK   Ext                 -  1302
=COMCK+  Abs   12974 #032AE -  1856
 COMCK1  Ext                 -  1711  1856
=COMCK0  Abs   12970 #032AA -  1855  1785  1921  2037
 CONCOM  Ext                 -  1543
=CRGJMP  Abs   12410 #0307A -  1194
=D1C=R3  Abs   12359 #03047 -  1115   815
 DATACK  Ext                 -  1678
=DECP    Abs   12943 #0328F -  1780  1786
 DECP00  Abs   12959 #0329F -  1785  1783
 DECP01  Abs   12966 #032A6 -  1788  1712  1739  1925  1929
 DECPE5  Abs   12916 #03274 -  1729  1781
=DEFP    Abs   12435 #03093 -  1260
 DEFP50  Abs   12521 #030E9 -  1295  1303
 DEFP70  Abs   12555 #0310B -  1308  1291
 DEFPE+  Abs   12414 #0307E -  1251  1267
 DEFPE6  Abs   12427 #0308B -  1256  1271
 DFKEYP  Ext                 -  1262
 DFPEXP  Abs   12431 #0308F -  1258  1299
=DIMP    Abs   12987 #032BB -  1905  1922
 DIMP05  Abs   13000 #032C8 -  1910  1907
 DIMP15  Abs   13021 #032DD -  1921  1911  1916  1938
 DIMP20  Abs   13042 #032F2 -  1931
 DIMPE5  Abs   13076 #03314 -  1982
 DISPP   Ext                 -   829
 DSTRP0  Abs   13095 #03327 -  2036  2039
 DSTRP5  Abs   13173 #03375 -  2159  2038
=DSTRYP  Abs   13080 #03318 -  2030
=DSTp    Abs   12890 #0325A -  1718  1705
 EOLCK   Ext                 -   779
 EOLCK+  Ext                 -   710
 ERR-TR  Abs   12141 #02F6D -   850   842
=ERR01   Abs   11819 #02E2B -   699  1251  1517  1700  2050  2260
=ERR02   Abs   11829 #02E35 -   703  1514
=ERR03   Abs   11839 #02E3F -   707   407
=ERR04   Abs   11868 #02E5C -   719   712
=ERR05   Abs   11878 #02E66 -   723  1729  1982
=ERR06   Abs   11888 #02E70 -   727  1256
=ERR08   Abs   11905 #02E81 -   736  1258
=ERR09   Abs   11915 #02E8B -   740
=ERR10   Abs   11925 #02E95 -   744
=ERR11   Abs   12034 #02F02 -   795
=ERR3    Abs   11858 #02E52 -   715
 ERR3*   Abs   11848 #02E48 -   710   708
=ERRDSP  Abs   12091 #02F3B -   825
 ERRIDS  Abs   12135 #02F67 -   849   823
```

```
 ERRIF    Abs   11935 #02E9F -   757   792
 ERRIF*   Abs   11954 #02EB2 -   765   819
 ERRIF+   Abs   11994 #02EDA -   779   775
 ERRIF1   Abs   11950 #02EAE -   764   760
 ERRIF2   Abs   12005 #02EE5 -   783   772
 ERRIF3   Abs   12014 #02EEE -   786   780
 ERRIF7   Abs   12020 #02EF4 -   787
 ERRIF9   Abs   12024 #02EF8 -   789   778   784
=ERRX0    Abs   12040 #02F08 -   804   700   704   716   720   724   737   741
                                 745
=ERRX10   Abs   12049 #02F11 -   808   805
 ERRX11   Abs   12052 #02F14 -   809   790
=ERRX20   Abs   12112 #02F50 -   841   524   814
=EXCHRe   Abs   11905 #02E81 -   735
 EXPPR+   Abs   12922 #0327A -  1733  1402  1718
 EXTIF+   Ext                 -  1044
 FIXP     Ext                 -   381  2166
=FIXp     Abs   13187 #03383 -  2166
=FNP      Abs   12600 #03138 -  1387
=FNP+     Abs   12600 #03138 -  1388
 FREE03   Abs   13212 #0339C -  2243  2241
 FREE05   Abs   13226 #033AA -  2249  2245
=FREE07   Abs   13245 #033BD -  2259  2164  2303
 FREE08   Abs   13248 #033C0 -  2260  2247
=FREE15   Abs   13235 #033B3 -  2255
=FREEp    Abs   13196 #0338C -  2238
=FSPECe   Abs   12034 #02F02 -   794
=GNXCR+   Abs   12391 #03067 -  1186  1552
 GNXTC1   Abs   12397 #0306D -  1188  1191
=GNXTCR   Abs   12388 #03064 -  1185  1528  2088  2238  2300
=IFCK     Abs   12418 #03082 -  1253  1269
=ILCNTe   Abs   11888 #02E70 -   726
 INADDR   Ext                 -   445   504   999
 INPTER   Abs   12851 #03233 -  1700  1692
 INPTP1   Abs   12816 #03210 -  1685  1680
 INPTP2   Abs   12835 #03223 -  1694  1689
=INPUTP   Abs   12794 #031FA -  1676
=IVEXPe   Abs   11829 #02E35 -   702
=IVPARe   Abs   11839 #02E3F -   706
=IVVARe   Abs   11878 #02E66 -   722
 LBLINP   Ext                 -   771
 LEXPTR   Ext                 -  1446
=LINE#?   Abs   12279 #02FF7 -  1051  1042
=LINPTP   Abs   12791 #031F7 -  1675
 LNEP26   Ext                 -  1054
 LNEP66   Ext                 -   831
 LNEPLN   Ext                 -  1053
 LNP55    Ext                 -   781
 LNPTP5   Abs   12935 #03287 -  1738  1710
=LPRNCK   Abs   13252 #033C4 -  2300  2251
 MAIN10   Ext                 -   847
 MEMERR   Ext                 -   331
 MFERR-   Ext                 -   846
=MSPARe   Abs   11868 #02E5C -   718
 NUMCK    Ext                 -  2255
```

```
 NUMCKO  Ext                  -  1933
 Ntoken  Ext                  -   128  1390  1542  1855
=OAGNXT  Abs   12384 #03060  -  1184  1305  1547
=OFFP    Abs   13157 #03365  -  2153
 OFFP10  Abs   13177 #03379  -  2163  2157
=OPTP    Abs   11598 #02D4E  -   379
 OPTP10  Abs   11630 #02D6E  -   389   383
 OPTP20  Abs   11649 #02D81  -   397   385
 OPTP30  Abs   11691 #02DAB  -   407   387   395
=OUT1T+  Abs   11487 #02CDF  -   263
=OUT1TK  Abs   11499 #02CEB  -   268   264  1184  1677  1713  1858
=OUT2TC  Abs   11517 #02CFD  -   274
=OUT2TK  Abs   11519 #02CFF  -   275
=OUT3TC  Abs   11538 #02D12  -   281
=OUT3TK  Abs   11541 #02D15  -   282
=OUTBY+  Abs   11493 #02CE5  -   266  1294  1401  1540
=OUTBYT  Abs   11496 #02CE8  -   267   501   588   828
 OUTNB4  Ext                  -  1273
 OUTNBS  Ext                  -   178
=OUTNIB  Abs   11560 #02D28  -   288  1286  1393
=OUTOVF  Abs   11592 #02D48  -   331   328
 OUTVAR  Ext                  -  1527
 OVFLCK  Abs   11579 #02D3B  -   326   269   276   283   289
=PARERR  Abs   12040 #02F08  -   803
 PILP    Ext                  -  1702
 PRENCK  Ext                  -  1304  2256
=PRNEXe  Abs   11925 #02E95  -   743
=QUOEXe  Abs   11915 #02E8B  -   739
 R.STPR  Ext                  -  1011
 R3=D1*  Ext                  -  1389
=READP   Abs   12855 #03237  -  1702
 READP2  Abs   12871 #03247  -  1709  1706
 READP4  Abs   12883 #03253  -  1713  1698
=READP5  Abs   12859 #0323B  -  1705  1682  1687  1714
=REDPE5  Abs   12907 #0326B  -  1726  1708  1722  1740
 REL3DO  Ext                  -   182
 REMP    Ext                  -   589
 RESCAN  Ext                  -   152  1037
=RESPTR  Abs   12658 #03172  -  1446   125   150   579   709   764   787   806
                             1508  1788  2034  2159
=REST*   Abs   12341 #03035  -  1071
 RESTA1  Abs   12165 #02F85  -   996   988
 RESTA2  Abs   12204 #02FAC  -  1009  1004
 RESTA3  Abs   12207 #02FAF  -  1010   997
 RESTA5  Abs   12243 #02FD3  -  1032
 RESTA7  Abs   12290 #03002  -  1054  1052
 RESTAR  Abs   12148 #02F74  -   987   811
 RESTxx  Abs   12170 #02F8A  -   999  1074
 RS-R03  Ext                  -  1253
 RSTR?5  Abs   12336 #03030  -  1068  1064
 RSTRT?  Abs   12296 #03008  -  1056   810
 RTNCC   Abs   13243 #033BB  -  2257   391   393   399   401   403   405  2250
 S-R0-0  Ext                  -   508   767
 S-R1-3  Ext                  -  1057  1071
=SHOWp   Abs   13193 #03389  -  2235
```

```
 SPLVRP  Abs   13067 #0330B -  1978  1295  2036
 STLXP2  Ext               -  1017
 STMTDO  Ext               -  1012
=STMTL+  Abs   11716 #02DC4 -   500
 STMTL2  Abs   11753 #02DE9 -   510   507
=STMTLO  Abs   11719 #02DC7 -   501
 STMTLe  Abs   11778 #02E02 -   523   512
 STRGCK  Ext               -  1695
 SVRST2  Ext               -  1039
=SYNTXe  Abs   11819 #02E2B -   698
=TRACEP  Abs   13109 #03335 -  2041
 TRCPE1  Abs   13138 #03352 -  2050  1919  1936
 TRNFCK  Ext               -   822   841
=UPDIN+  Abs   11698 #02DB2 -   445   769
=UPDINA  Abs   11695 #02DAF -   443
 USFPE2  Abs   12678 #03186 -  1510  1409
 USRFE+  Abs   12691 #03193 -  1516  1399  1550
 USRFE1  Abs   12694 #03196 -  1517  1537  1544
 USRFP   Abs   12698 #0319A -  1519  1282  1391
 USRFP2  Abs   12720 #031B0 -  1527  1525
 USRFP7  Abs   12673 #03181 -  1507  1406
 USRFPE  Abs   12684 #0318C -  1513  1404
 VARP    Ext               -  1780  1906  1978
 WRDS10  Abs   11315 #02C33 -   129   153
 WRDS20  Abs   11329 #02C41 -   134   173
 WRDS35  Abs   11364 #02C64 -   146  2155
 WRDS40  Abs   11366 #02C66 -   149   141
 WRDS50  Abs   11385 #02C79 -   156   139
 WRDS55  Abs   11414 #02C96 -   168   158   163   166
=WRDSC+  Abs   11302 #02C26 -   125
=WRDSCN  Abs   11306 #02C2A -   126   379   389   397  1260  1685  2030  2041
                           2153  2243
=XCHECK  Abs   11460 #02CC4 -   184
 XCHEK1  Abs   11464 #02CC8 -   185   140
 eEXCHR  Ext               -   736   786
 eFSPEC  Ext               -   795
 eILCNT  Ext               -   727
 eILEXP  Ext               -   703
 eILPAR  Ext               -   715
 eILVAR  Ext               -   723
 eL2LNG  Ext               -   523
 eMSPAR  Ext               -   719
 ePRNEX  Ext               -   744
 eQUOEX  Ext               -   740
 eSYNTX  Ext               -   699   783   849
 errif   Abs   12031 #02EFF -   792   826
 errx0   Abs   11911 #02E87 -   737   729   733
 exppar  Ext               -  1736
 lnep66  Abs   12108 #02F4C -   831  2032  2043  2045  2047
 ntken+  Ext               -   758
 rtncc   Abs   12920 #03278 -  1731  1724
 t!      Ext               -   587
 tALL    Ext               -  2031
 tANGLE  Abs  393651 #601B3 -    13   382
 tBASE   Ext               -   380
```

tCOMMA	Ext		-	1400	1688				
tDEGRE	Ext		-	390					
tDISP	Ext		-	827					
tERROR	Ext		-	2154					
tFLOW	Abs	2687471	#901EF	-	13	2046			
tFN	Ext		-	1265					
tKEY	Ext		-	1261					
tNEAR	Abs	3932655	#C01EF	-	13	398			
tNEG	Abs	3998191	#D01EF	-	13	402			
tOFF	Ext		-	2042					
tPORT	Ext		-	2244					
tPOS	Abs	4325811	#201B3	-	13	400			
tPRMST	Ext		-	1293					
tRDIAN	Ext		-	392					
tROUND	Abs	4981231	#C01EF	-	13	384			
tSEMIC	Ext		-	1539	1545	1686	1696	1931	
tSVAR	Ext		-	1523					
tTIMER	Ext		-	2156					
tVARS	Abs	5964271	#B01EF	-	13	2044			
tXFN	Ext		-	160	187				
tXWORD	Ext		-	137	184				
tZERO	Abs	1835503	#C01EF	-	13	404			

Input Parameters

  Source file name is JP&PR2::MS

  Listing file name is JP/PR2:TI:ML::-1

  Object file name is JP%PR2:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                    STITLE PIL Parse - # Parse
                    TITLE  Parse Routines - Part 3<831212.1204>
033D6               ABS    #033D6
                *       J  PPPP    &     PPPP   RRRR    333
                *       J  P   P  & &    P   P  R   R  3   3
                *       J  P   P  & &    P   P  R   R      3
                *       J  PPPP    &     PPPP   RRRR     33
                *       J  P      & & &  P      R R        3
                *  J    J  P      &  &   P      R  R   3   3
                *   JJJ   P       && &  P      R   R   333
                      RDSYMB TI%EQU::MS
               ************************************************************
               ************************************************************
               **
               ** Name:    PILP    -  Channel Number, Record Number Parse
               **
               ** Category:   PARUTL
               **
               ** Purpose:    Parses Channel Number and Record Number
               **
               **            PILP entry parses both the channel number
               **            and the optional record number.  Channel
               **            number must either be followed by a comma
               **            and a record number, or it must be
               **            immediately followed by a semi-colon; other-
               **            wise an error exit is taken.  If the channel
               **            number is followed by a comma and a record
               **            number, then this entry makes a distinction
               **            as to whether or not the record number is
               **            followed by a semi-colon - if it isn't this
               **            routine assumes that statement parse is
               **            complete (ie no parameter list) and returns
               **            directly to the main parse driver.  If it
               **            is followed by a semi-colon, it returns to
               **            the caller.  This entry point is utilized
               **            by both READ# and PRINT#.
               **
               **            PILP+ entry parses the channel number only.
               **            It indicates on exit by use of a status bit
               **            setting as to whether or not the channel
               **            number was followed by a comma.
               **
               ** Entry:     2 entry points:
               **            D(A) = (AVMEME)
               **            D1 points to optional blanks preceding #
               **            D0 points into output buffer
               **            1) PILP  - Caller must be on the same
               **                       subroutine level as if they
               **                       were called by the main parse
               **                       driver.
               **            2) PILP+ - S8=1
               **
               ** Exit:      Carry set =>
               **             No '#' found
               **            Carry clr =>
```

```
56          **              Found '#' and parsed channel#
57          **              P=0
58          **              PILP entry:
59          **               tSEMIC was last token found and output
60          **                (syntax was <channel#> [<record#>] ;
61          **               D1 past ascii ;
62          **               D0 past tSEMIC
63          **
64          **              PILP+ entry:
65          **               S8=1 => No comma after channel#
66          **               S8=0 => tCOMMA found and output
67          **
68          **             No rtn to caller (only to main parse driver) =>
69          **             (only for PILP entry)
70          **              Record number wasn't followed by a semi-colon
71          **
72          **             Error exits if neither ',' or ';' found after
73          **             channel number (PILP entry only)
74          **
75          ** Calls:    #CK, OUT1TK, NUMCK, RESPTR, COMCK+, NUMC+0
76          **
77          ** Uses:     A-C,D(15-5),S0-S3,S7,S8,S11,R0,R1,R3,FUNCD0
78          **
79          ** Stack lvls: 5
80          **
81          ** Detail:     For PILP entry, if there's no terminating semi-
82          **             colon after the record#, a level is popped off
83          **             the return stack before returning.  This is
84          **             because it assumes the statement syntax is
85          **             complete.
86          **
87          **             Another assumption made by PILP entry is that
88          **             if there's no record number specified, there
89          **             had better be a variable or expression list
90          **             (which is always preceded by a semi-colon).
91          **
92          **          Acceptable syntax for commands calling PILP/PILP+
93          **
94          **             PRINT# uses same general syntax as READ#:
95          **          READ #<channel no.>; <vbl/expr list>
96          **          READ #<channel number>, <record number>
97          **          READ #<channel no.>, <record no.>; <vbl/expr list>
98          **
99          **             RESTORE #<channel number> [,<record number>]
100         **
101         ** History:
102         **
103         **    Date     Programmer   Modifications
104         **  --------   ----------   -------------
105         **  07/08/82   S.W.         Modified documentation
106         **  05/11/83   S.W.         Replaced call to COMCK1 w/COMCK+
107         **
108         ****************************************************************
109         ****************************************************************
110         *
```

```
111 033D6 848  =PILP   ST=0   8
112 033D9 7000 =PILP+  GOSUB  =#CK
113 033DD 400          RTNC                  # not found?
114 033E0 72B2         GOSUB  NUMC+0         WILL BE SAVED IN R3
115 033E4 7000         GOSUB  =COMCK+
116 033E8 441          GOC    PILP30         COMMA AFTER CHANNEL #?
117             * NO COMMA FOUND AFTER CHANNEL#
118 033EB 878          ?ST=1  8              CALLED BY RESTORE#?
119 033EE 32           GOYES  ResPtr
120             * NO COMMA, BUT CALLED BY READ#/PRINT#
121 033F0 3100         LC(2)  =tSEMIC
122 033F4 966          ?A#C   B              NO EXPR/VAR LIST?
123 033F7 32           GOYES  PILP85
124 033F9 6000  PILP10 GOTO   =OUT1TK        OUTPUT SEMICOLON TOKEN
125             * COMMA AFTER CHANNEL #
126 033FD 878   PILP30 ?ST=1  8              CALLED BY RESTORE#?
127 03400 51           GOYES  PILP80
128             *
129 03402 7792         GOSUB  NUMCK          PARSE RECORD#
130 03406 3100         LC(2)  =tSEMIC
131 0340A 962          ?A=C   B              TERMINATING SEMI-COLON?
132 0340D CE           GOYES  PILP10
133             *
134 0340F 07    PILP40 C=RSTK                RETURN TO MAINLOOP
135 03411 6000  ResPtr GOTO   =RESPTR        BACK PTR UP
136             *
137 03415 848   PILP80 ST=0   8              INDICATES COMMA AFTER CHANNEL #
138 03418 03           RTNCC
139             *
140 0341A 4F6   PILP85 GOC    FORPE1         (B.E.T.)
141             *
```

```
142                     STITLE FOR/NEXT Parse
143                 ********************************************************************
144                 ********************************************************************
145                 **
146                 ** Name:    FORP    -  FOR Statement Parse
147                 **
148                 ** Category:   STPARS
149                 **
150                 ** Purpose:    Parses FOR Statement
151                 **
152                 ** Entry:      D(A) = (AVMEME)
153                 **             D1 past FOR keyword
154                 **             D0 points into output buffer past tFOR
155                 **
156                 ** Exit:       Statement properly parsed =>
157                 **              Return with carry clear
158                 **              P=0
159                 **              D1 past syntactically correct FOR statement
160                 **              D0 past tokenized statement in output buffer
161                 **
162                 **             Else error exit to PARERR
163                 **
164                 ** Calls:      NXTP, GNXTCR, NUMCK, WRDSCN, RESPTR
165                 **
166                 ** Uses:       A-C,D(15-5),S0-S3,S7,S11,FUNCD0,D0,D1,R0,R1,R3
167                 **
168                 ** Stk lvls: 5
169                 **
170                 ** History:
171                 **
172                 **    Date     Programmer   Modifications
173                 ** ----------  ----------   -------------
174                 **  10/21/82   S.W.         Wrote documentation
175                 **
176                 ********************************************************************
177                 ********************************************************************
178                 *
179                 *
180 0341D 7430 =FORP   GOSUB  NXTP
181 03421 7000        GOSUB  =GNXTCR
182 03425 31D3        LCASC  \=\
183 03429 966         ?A#C   B
184 0342C 73          GOYES  FORPER
185 0342E 3100        LC(2)  =tCOMMA
186 03432 DA          A=C    A
187 03434 7E52        GOSUB  NUMC+0          NUMERIC EXPR S/FOLLOW
188 03438 3100        LC(2)  =tTO
189 0343C 966         ?A#C   B               NOT T_TO?
190 0343F B4          GOYES  FORPE1
191 03441 7452        GOSUB  NUMCK0
192 03445 8E00        GOSUBL =WRDSC+         RESTORE POINTER FIRST
          00
193 0344B 00          CON(2) =tSTEP
194 0344D 000         REL(3) =FIXp           T_STEP?
195 03450 00          CON(2) 0
```

```
196 03452 5EB          GONC   ResPtr          (B.E.T.)
197              *
198              ***************************************************************
199              ***************************************************************
200              **
201              ** Name:(S) NXTP    -  NEXT statement parse
202              **
203              ** Category:   PARUTL
204              **
205              ** Purpose:
206              **      Parses NEXT Statement.  Also useful for
207              **      simple numeric variable parse.
208              **
209              ** Entry:
210              **      D(A) = (AVMEME)
211              **      D1 at alleged simple numberic variable
212              **      D0 points into output buffer
213              **
214              ** Exit:
215              **      Carry clear =>
216              **       Simple numeric variable found and output
217              **       P      =  0
218              **       Carry clear
219              **       D1 advanced past variable
220              **       D0 points past tokenized variable
221              **
222              **      Else error exit to PARERR with eILVAR
223              **
224              ** Calls:      VARP
225              **
226              ** Uses.......
227              **      A-C, D0, D1, S0-S3, S11, R0
228              **
229              ** Stk lvls:   4
230              **
231              ** NOTE:
232              **      This also serves as parse for NEXT statement
233              **
234              ** History:
235              **
236              **    Date     Programmer              Modification
237              **  --------   ----------   ----------------------------------
238              **  02/03/83   S.W.         Added documentation
239              **
240              ***************************************************************
241              ***************************************************************
242              *
243 03455 75B0 =NXTP   GOSUB  VARP
244 03459 5B5          GONC   FORPE5          STRING VARIABLE?
245 0345C 872          ?ST=1  2               ARRAY VAR?
246 0345F 65           GOYES  FORPE5
247 03461 03           RTNCC
248              *
249 03463 6000  FORPER GOTO   =FREE07         Set S4; ERR01
250              *
```

```
                STITLE IF Parse
*****************************************************************
*****************************************************************
**
** Name:    IFP     -  IF statement parse
** Name:    ELSEP   -  ELSE parse
**
** Category:   STPARS
**
** Purpose:    IFP entry parses the first part of an IF
**             construct (up through THEN).  If THEN is
**             immediately followed by a line number or
**             a quoted label, the parse continues
**             through the implied GOTO.  It also traps
**             out implied LET and transfers control to
**             the appropriate parse with S10 and S9 set,
**             flagging this as an implied LET in the
**             'middle' of an IF statement; this prevents
**             a failed implied LET parse from being re-
**             parsed as implied DISP - instead it will
**             be parsed as a label (implied GOTO).
**
**             ELSEP entry parses what immediately
**             follows the ELSE keyword.  This is
**             analogous to what is said for THEN above.
**             This entry is used by the main parse
**             driver when it encounters the ELSE key-
**             word and it has been determined that it
**             is appropriate (S6=1)
**
** Entry:      S6=0
**             D(A) = (AVMEME)
**             IFP entry:
**              D1 is past IF
**              D0 points into output buffer past tIF
**             ELSEP entry:
**              D1 is past ELSE
**              D0 points into output buffer past tTHEN (t@)
**
** Exit:       P=0
**             S6=1 => ELSE is a legal stmt terminator
**
**             Returns to main driver with carry clr =>
**              THEN/ELSE was followed by a line number
**               or a quoted label
**              D1 is past the line number or label
**              D0 is past the tokenized implied GOTO
**
**             Exits via LNP06 =>
**              This looks like an 'extended IF' of the
**               type that THEN/ELSE is followed
**               immediately by an implied LET.
**              S10=S9=1 (Don't try implied DISP if errors)
**              A contains variable or user-defined
**               function tokenization.
```

```
306             **              D1 is advanced past the corresponding text.
307             **              tEXTIF (t@) has been output
308             **              D0 is 4 nibbles beyond tEXTIF
309             **               (beyond statement length byte)
310             **
311             **             Returns to main driver with carry set =>
312             **              This looks like an 'extended IF' but
313             **               it is not an implied LET; ie it looks
314             **               like THEN/ELSE is immediately followed
315             **               by a begin BASIC keyword.
316             **              S10=S9=1 (Don't try implied DISP if errors)
317             **              A = The token following THEN/ELSE
318             **              D1 is advanced past the corresponding text
319             **              tEXTIF (t@) has been output
320             **              D0 is 4 nibbles beyond tEXTIF
321             **               (beyond statement length byte)
322             **
323             ** Calls:     NUMCK, WSR0-3, LBLNIF, WRDSCN, SVRSTA
324             **            eolck+, FINDA, R.STPR, STMTLO, UPDIN+
325             **
326             ** Uses:        A-C,D(15-5), D1,D0, R0,R1,R3, S0-S3,S6,S7,S9,
327             **              FUNCD0, S-R0-3, S-R1-0, S-R1-2, S-R1-3
328             **
329             ** Stk lvls:    6
330             **
331             ** Detail:      'IF' illegal after 'pending' THEN
332             **              S6 is the global flag that indicates whether
333             **              or not ELSE is a valid statement terminator.
334             **              It also indicates whether or not IF is a
335             **              legal construct in the current context:
336             **              S6=0 => IF is a legal construct
337             **                      ELSE is not a legal terminator
338             **                      Either we're not currently parsing
339             **                       within an IF construct OR
340             **                       We are, but it's after the ELSE.
341             **
342             **              S6=1 => IF is not a legal construct
343             **                      ELSE is a legal statement terminator
344             **                      We are parsing within an IF
345             **                       construct, but we are prior to ELSE.
346             **
347             ** Algorithm:
348             **
349             **            Set IF in progress flag (S-R0-3);
350             **            Parse numeric expression;
351             **            If not tTHEN
352             **              Error
353             **            Clear RESTART flag (S-R1-3); Clear Err# (S-R1-0);
354             **            Call NTOKEN; Set RESTART flag if XWORD/XFN;
355             **            Save RESTART address (S-R1-2);
356             **            Save contents of LEXPTR in STMTD0;
357             **
358             **              If variable (letter not associated with token)
359             **              or user-defined function name
360             **                Call EXTIFP;
```

```
361            **              Goto C2 (parse driver - see algorithm JP&PR1).
362            **
363            **            If single or double quote
364            **              Call LBLNIF;
365            **              If legal label
366            **                THEN return with carry clear
367            **                ELSE error
368            **            If line#
369            **              Call LBLINF; return with carry clear
370            **
371            **   EXTIFP:  Save token;
372            **            Set S9  (Middle of IF) & S10 (Implied LET)
373            **             (Needed so string functions & failed assignments
374            **             don't parse as implied DISP, but as labels)
375            **            Output tEXTIF (looks like @);
376            **            Calculate & write out stmt length in previous
377            **            INADDR; Update INADDR; Restore token;
378            **            Return with carry set
379            **
380            ** History:
381            **
382            **    Date     Programmer   Modifications
383            **  --------   ----------   -------------
384            **  10/21/82   S.W.         Wrote documentation
385            **  01/06/83   S.W.         Added algorithm
386            **
387            ********************************************************************
388            ********************************************************************
389            *
390            *
391 03467 137  =IFP    CD1EX                Save D1
392 0346A AC2         C=0    S
393 0346D A4E         C=C-1  S             Set Global IF flag
394 03470 8E00        GOSUBL =WSR0-3        in S-R0-3
        00
395 03476 135         D1=C                 Restore D1
396 03479 7022        GOSUB  NUMCK         SEE IF NUMERIC EXPR
397 0347D 8E00        GOSUBL =WRDSC+       RESTORE PTR FIRST
        00
398 03483 00          CON(2) =tTHEN
399 03485 900         REL(3) IFP50         NOT T_THEN?
400 03488 00          CON(2) 0
401            *
402 0348A 6000  FORPE1 GOTO   =ERR01
403            *
404 0348E 856   IFP50  ST=1   6             PENDING THEN FLAG (ELSE LEGAL)
405            *
406            *   Will be used if Not Implied LET statement
407            *
408 03491 7267 =ELSEP  GOSUB  eolck+        SOME command MUST follow THEN/ELSE
409 03495 471         GOC    KYWMSG        End of statement token?
410 03498 8E00        GOSUBL =SVRST+       SAVE RESTART ADDRESS & SET FLAG
        00
411 0349E 86B         ?ST=0  11            NOT A VARIABLE?
412 034A1 81          GOYES  ELSEP5
```

```
413                *
414                * MUST BE IMPLIED LET OR LABEL
415                * SET FLAGS
416                *
417 034A3 7E20   ELSEP2 GOSUB  EXTIFP
418                * S9 set => Middle of IF
419 034A7 8C00          GOLONG =LNP06          Try Implied LET parse
          00
420                * Key word missing
421 034AD 7000   KYWMSG GOSUB  =!CK            Only ! legal as stmt terminator her
422 034B1 6550          GOTO   ELSEP7          Rtn if ! found, else give syntax er
423                *
424 034B5 6000   FORPE5 GOTO   =ERR05
425                *
426                * MUST BE LINE#, QUOTED LABEL, OR KEYWORD.
427                * HAVE ALREADY TRAPPED OUT IMPLIED LET, LABEL NAMES WHICH EITHER
428                * 1) START WITH A LETTER OR 2) ARE A STRING VARIABLE
429                *
430                * Assumes:
431                *   Maximum Level of Subroutines used
432                *     LINEP
433                *     IF
434                *     LBLNIF
435                *     LABELP
436                *     EXPPAR
437                *     NTOKEN
438                *     NTOKEN (uses 2 level internally)
439                *
440                * If line# or quote
441                *    Gosub to Label | Line# parse
442                * else
443                *    Return SC  (Must be extended IF with BASIC keyword)
444                *
445                *
446 034B9 8E00   ELSEP5 GOSUBL =FINDA
          00
447 034BF 00            CON(2) =tFN
448 034C1 2EF           REL(3) ELSEP2
449 034C4 00            CON(2) =tLINE#         Line#
450 034C6 830           REL(3) ELSEP6          Yes
451 034C9 22            NIBASC \"\             Quoted label ?
452 034CB 330           REL(3) ELSEP6
453 034CE 72            NIBASC \'\             Other quote ?
454 034D0 E20           REL(3) ELSEP6
455 034D3 00            CON(2) 0
456                *
457 034D5 85A    EXTIFP ST=1   10              So ERRDSP not done
458 034D8 A4C           A=A-1  S               & S-R0-0 written to
459 034DB 480           GOC    EXTIF+          Not an XWORD or XFN?
460 034DE 8E00          GOSUBL =R.STPR         Write out E instead of F
          00
461                *
462 034E4 859   =EXTIF+ ST=1   9               Middle of IF
463 034E7 101           R1=A                   Save token
464 034EA 3100          LC(2)  =tEXTIF         Extended IF token
```

```
465                * Only write out previous INADDR 1st time through (S10=1)
466                *  From RESTART, it isn't updated
467 034EE 7000             GOSUB  =STMTLO        Outbyt byte & calc stmt length
468 034F2 7000             GOSUB  =UPDIN+
469 034F6 111              A=R1                  Restore the token
470 034F9 85A              ST=1   10             Set for RESTART
471 034FC 02               RTNSC
472                *
473 034FE 849    ELSEP6    ST=0   9              Allow line# or label
474 03501 8E00             GOSUBL =LBLNIF
            00
475 03507 500    ELSEP7    RTNNC
476 0350A 66B5             GOTO   ASNPE+         Syntax error; Set S4
477                *
478                *
```

```
479                     STITLE Variable Parse & Output
480             ********************************************************************
481             ********************************************************************
482             **
483             ** Name:(S) VARP    -  Variable Parse
484             ** Name:    VARP05  -  Variable Parse
485             **
486             ** Category:   PARUTL
487             **
488             ** Purpose:
489             **      Checks for a variable token.  If found, it is output; if
490             **      the token is not a variable token, an error exit is
491             **      taken.
492             **
493             **      VARP entry assumes that D1 points to optional blanks
494             **      preceding the text to check.
495             **
496             **      VARP05 entry assumes that NTOKEN has already been called,
497             **      and that the token to check is in register A.
498             **
499             ** Entry:
500             **      D(A) = (AVMEME)
501             **      D0 points into the output buffer
502             **      2 entry points:
503             **      1) VARP   - D1 at optional blanks preceding text to
504             **                  be examined.
505             **      2) VARP05 - Register A contains alleged variable token.
506             **                  D1 points past the corresponding text as per
507             **                  NTOKEN exit.
508             **                  (LEXPTR) as per NTOKEN exit.
509             **
510             ** Exit:
511             **      Return to caller =>
512             **      Variable parsed
513             **      Tokenized variable written to output buffer
514             **      D0 past variable tokenization in output buffer
515             **      D1 past variable name
516             **      Carry set =>
517             **            Numeric variable found
518             **      Carry clr =>
519             **            String variable found
520             **
521             **      Error exit if variable not found or if MEMERR
522             **
523             ** Calls:      NTOKEN, OUTVAR
524             **
525             ** Uses.......
526             **  Exclusive: A,D0,D1
527             **  Inclusive: A,B,C,S0-S3,S11,D0,D1,R0
528             **
529             ** Stk lvls:   3
530             **
531             ** History:
532             **
533             **    Date     Programmer             Modification
```

```
534                ** --------  ----------  ---------------------------------
535                ** 07/06/82   JP          Modified documentation
536                **
537                ********************************************************************
538                ********************************************************************
539                *
540                *
541 0350E 7173 =VARP   GOSUB  ntoken
542 03512 86B  =VARP05 ?ST=0  11            NOT VARIABLE?
543 03515 0A           GOYES  FORPE5        ILLEGAL VARIABLE
544 03517 7322         GOSUB  OUTVAR
545 0351B 861          ?ST=0  1             NUMERIC?
546 0351E 00           RTNYES
547 03520 01           RTN
548                *
549                *
```

```
                        STITLE Save D0/C and D1/A in R3  (R3=D1C)
                    ********************************************************************
                    ********************************************************************
                    **
                    ** Name:(S) R3=D10  -  Save D0 and D1 in R3
                    ** Name:    R3=D1C  -  Save C(A) & D1 in R3
                    ** Name:    R3=D1+  -  Save C(A) & A(A) in R3
                    **
                    ** Category:   GENUTL
                    **
                    ** Purpose:
                    **      R3=D10 entry saves D0 in R3(A) and D1 in R3(9-5).
                    **      R3=D1C entry saves C(A) in R3(A) and D1 in R3(9-5).
                    **      R3=D1+ entry saves C(A) in R3(A) and A(A) in R3(9-5).
                    **
                    ** Entry:
                    **      R3=D10:  D0 and D1 contain values to save in
                    **               R3(A) and R3(9-5), respectively.
                    **      R3=D1C:  C(A) and D1 contain values to save in
                    **               R3(A) and R3(9-5), respectively.
                    **      R3=D1+:  C(A) and A(A) contain values to save in
                    **               R3(A) and R3(9-5), respectively.
                    **
                    ** Exit:
                    **      Carry preserved from entry
                    **      A(A)=C(A)
                    **      R3=D10: R3(A)=D0 on entry;   R3(9-5)=D1 on entry
                    **              C(A)=A(A)=D0
                    **      R3=D1C: R3(A)=C(A) on entry; R3(9-5)=D1 on entry
                    **      R3=D1+: R3(A)=C(A) on entry; R3(9-5)=A(A) on entry
                    **
                    ** Calls:      None
                    **
                    ** Uses.......
                    **      R3=D10: A, C(A), R3
                    **      R3=D1C: A, R3
                    **      R3=D1+: A, R3
                    **
                    ** Stk lvls:   0
                    **
                    ** History:
                    **
                    **    Date     Programmer             Modification
                    **   --------   ----------   ----------------------------------
                    **   07/06/82   JP           Modified documentation
                    **
                    ********************************************************************
                    ********************************************************************
                    *
                    *
03522 7000  =R3=D1* GOSUB  =RESPTR
03526 136   =R3=D10 CDOEX               Save D1,D0
03529 134           D0=C
                    *
0352C 133   =R3=D1C AD1EX               Save D1,C
```

```
    605 0352F 131           D1=A
    606 03532 BF0  =R3=D1+  ASL    W              Save A,C
    607 03535 BF0           ASL    W
    608 03538 BF0           ASL    W
    609 0353B BF0           ASL    W
    610 0353E BF0           ASL    W
    611 03541 DA            A=C    A
    612 03543 103           R3=A
    613 03546 01            RTN
```

```
614                 STITLE REM Parse
615             ********************************************************************
616             ********************************************************************
617             **
618             ** Name:    REMP    -  REM Statement Parse and ! Parse
619             ** Name:    REMP10  -  Entry for DATA, REM and ! Decompile
620             **
621             ** Category: STPARS
622             **
623             ** Purpose:  REMP entry parses the REM statement and comments
624             **           beginning with '!'.  It does this by copying the
625             **           ascii characters in the input stream to the output
626             **           buffer until D0 (endline) is encountered.
627             **
628             **           REMP10 is an entry point for decompile; it
629             **           decompiles DATA, REM and !, by copying the ascii
630             **           characters from the token stream to the output
631             **           buffer.
632             **
633             ** Entry:    D(A) = (AVMEME)
634             **           D0 points into output buffer
635             **           P=0
636             **           2 entry points:
637             **           1) REMP   - D1 past REM keyword or '!'
638             **                       D0 past tREM or t!
639             **           2) REMP10 - D1 past tREM, tDATA or t!
640             **                       D0 past REM/DATA keyword or '!',
641             **                        followed by ascii blank
642             **
643             ** Exit:
644             **           Carry clear
645             **           D1 at 0D (Endline)
646             **           D0 points into output buffer
647             **
648             **           REMP:       Ascii remark in output buffer
649             **                       D0 past output 0D
650             **           REMP10:     D0 past last ascii character
651             **
652             ** Calls:    BLKOK, CHK-0D
653             **
654             ** Uses:     A(B),C(B),D0,D1
655             **
656             ** Detail:   Remarks and DATA parse are terminated with
657             **           tokenization D0OF.
658             **           i.e.   tREM|t!|tDATA <ascii stream> D0OF
659             **
660             ** Stk lvls: 3
661             **
662             ** History:
663             **
664             **    Date     Programmer   Modifications
665             **  --------   ----------   -------------
666             **  10/21/82   S.W.         Added special header
667             **  04/29/83   S.W.         Added call to BLKOK
668             **  05/02/83   S.W.         Added call to CHK-0D
```

```
669             **
670             ********************************************************************
671             ********************************************************************
672             *
673 03548 8E00  =REMP   GOSUBL =BLKOK
          00
674 0354E 550          GONC   REMP10          Carry set => blank
675 03551 171  =REMP05 D1=D1+ 2
676 03554 75D1 =REMP10 GOSUB  CHK-0D
677 03558 5BF          GONC   REMP10          (B.E.T.)
678             *
```

```
679                 $TITLE DATA Parse
680             ********************************************************************
681             ********************************************************************
682             **
683             ** Name:    DATP    -  DATA Statement Parse
684             **
685             ** Category: STPARS
686             **
687             ** Purpose:  Parses DATA statement
688             **
689             ** Entry:    D(A) = (AVMEME)
690             **           D1 points past DATA keyword
691             **           D0 points into output buffer, past tDATA
692             **
693             ** Exit:     Valid parse=>
694             **             Return with carry clear
695             **             D1 points past syntactically correct statement
696             **             D0 points past tokenized statement
697             **             P=0
698             **
699             ** Calls:    DATACK (OUTLIT), BLNKCK, OUT1T+, CHK-OD
700             **
701             ** Uses:     A-C, D0,D1
702             **
703             ** Detail:   Quoted strings may be delimited by single or double
704             **           quotes, allowing any characters (except of course
705             **           the particular delimiting quote) to be quoted.
706             **
707             **           Unquoted strings allow any characters except commas.
708             **
709             **           Quoted strings which are followed by extraneous non-
710             **           blank characters before the comma or endline are
711             **           treated as unquoted strings (i.e. everything from
712             **           the preceding comma to the following comma or
713             **           endline is considered an unquoted string).
714             **
715             **           On unquoted strings, leading and trailing blanks are
716             **           truncated.
717             **
718             **           Due to the parse algorithm, unquoted strings which
719             **           contain a quote are considered to contain a quoted
720             **           string and so there must be a matching quote.
721             **
722             **           At execution:
723             **           DATA <endline> is interpreted as
724             **           DATA ""
725             **
726             **           and DATA <string>,   is interpreted as
727             **           DATA <string>,""
728             **               This is different from the HP-85
729             **
730             ** Stack lvls: 4
731             **
732             ** History:
733             **
```

```
734             **    Date      Programmer   Modifications
735             **  --------    ----------   ------------
736             **  10/21/82    S.W.         Added Special header
737             **  04/28/83    S.W.         Allow unquoted strings to contain
738             **                            quoted characters
739             **  05/02/83    S.W.         Added call to CHK-0D
740             **
741             ********************************************************************
742             ********************************************************************
743             *
744             *
745 0355B 8E00  DATP00 GOSUBL =BLNKCK
          00
746 03561 181         D0=D0- 2
747 03564 75C1        GOSUB  CHK-0D
748             *
749 03568 7271 =DATP  GOSUB  DATACK
750 0356C 14B  DATP30 A=DAT1 B
751 0356F 31C2        LCASC  \,\
752 03573 962         ?A=C   B
753 03576 5E          GOYES  DATP00
754 03578 31D0        LCHEX  0D
755 0357C 962         ?A=C   B
756 0357F CD          GOYES  DATP00
757 03581 3122        LCASC  \"\
758 03585 962         ?A=C   B
759 03588 0E          GOYES  DATP
760 0358A 300         LC(1)  =a'
761 0358D 962         ?A=C   B
762 03590 8D          GOYES  DATP
763 03592 7091        GOSUB  out1t+        Preserve blnks in unquoted string
764 03596 55D         GONC   DATP30        (B.E.T.)
765             *
```

```
                    $TITLE PRINT/DISP Parse
                ********************************************************************
                ********************************************************************
                **
                ** Name:     PRTP    -  PRINT Statement Parse
                ** Name:(S)  DISPP   -  DISP Statement Parse
                ** Name:     DSPP02  -  Implied DISP Statement Parse
                ** Name:(S)  USINGp  -  USING statement Parse
                **
                ** Category: STPARS
                **
                ** Purpose:  PRTP parses the PRINT statement.
                **
                **           DISPP parses the DISP statement.
                **           It is also used to parse an implied DISP
                **           when implied LET parse has failed.
                **
                **           DSPP02 parses implied DISP.  The distinction
                **           between DSPP02 and DISPP is that with DSPP02
                **           entry, parse errors result in a return to the
                **           caller; this entry is used on an alleged
                **           implied DISP that cannot be an implied LET,
                **           ie one that doesn't start with a variable or
                **           user-defined function name.
                **
                **           USINGp parses USING part of PRINT USING stmt
                **           This entry point used by HPIL for ENTER USING
                **
                ** Entry:    D(A) = (AVMEME)
                **           D1 points at input stream
                **           D0 points into output buffer
                **           3 entry points:
                **           1) PRTP   - D1 past PRINT keyword
                **                       D0 past tPRINT
                **           2) DISPP  - D0 is past tDISP.
                **                       Either D1 is past the DISP keyword
                **                        OR
                **                       D1 is at the beginning of a statement
                **                       that failed implied LET parse and
                **           3) DSPP02 - D1 at alleged expression list that
                **                        doesn't start with a variable or
                **                        user-defined function name.
                **                       tDISP has been output and D0 points
                **                       past it.
                **                       S8=1
                **                       If needed, D1/D0 have been saved
                **                       somewhere so that in case of error
                **                       they can be recovered.
                **
                **           4) USINGp - D1 at USING keyword
                **
                ** Exit:
                **           Carry clear =>
                **            P=0
                **            D1 past syntactically correct statement
```

```
821             **          DO past tokenized statement in output buffer
822             **
823             **         Carry set (DSPP02 entry only) =>
824             **          Not a valid implied DISP statement
825             **
826             **         Else error exit of some kind:
827             **          To PARERR (PRTP, DISPP entry only)
828             **          or to MEMERR (possible for all entry points)
829             **
830             ** Calls:   EXPPAR, NTOKEN, OUT1TK, NUMCK, PILP, COMCK, WRDSCN,
831             **         LBLINP, EOLCKR, RESPTR, R3=D10, D1C=R3
832             **
833             ** Uses:    A-C,D(15-5),S0-S3,S7,S8,S9,S11,R0-R3,FUNCDO
834             **
835             ** NOTE:  No routines called may use S8 (except PILP), S9
836             **       No routines below DISPP entry point may use R3 -
837             **          See LNP00 utility
838             **
839             ** Detail:  The PRINT statement is tokenized identical to the
840             **         DISP statement, except for tPRINT instead of tDISP.
841             **         PRINT# is tokenized very differently from PRINT.
842             **
843             **         Compiled DISP statement looks like:
844             **        tDISP [tUSING <tLINE# line#> | <string expr>]
845             **          [tSEMIC <display list>]
846             **
847             **         Compiled PRINT# statement looks like:
848             **   tPRINT #<channel no.>[tCOMMA <rec no.>]tSEMIC<exprs>
849             **   tPRINT #<channel no.> tCOMMA <record no.>
850             **
851             ** Stk lvls: 5 (if PRINT# then 6)
852             **
853             ** History:
854             **
855             **    Date     Programmer   Modifications
856             **  --------   ----------   ------------
857             **  10/21/82   S.W.         Eliminated capability for
858             **                           DISP USING <lbl>
859             **  04/29/83   S.W.         Disallow TAB in PRINT/DISP USING
860             **  05/02/83   S.W.         Create USING subroutine for use
861             **                           by PRINT/DISP, ENTER/OUTPUT
862             **  05/11/83   S.W.         Replaced 1 call to COMCK1 w/COMCK+
863             **
864             ****************************************************************
865             ****************************************************************
866             *
867 03599 793E =PRTP   GOSUB  PILP
868 0359D 490        GOC    DSPP02         NOT PIL COMMAND?
869             *
870 035A0 6E60       GOTO   PRT#P
871             *
872             *
873 035A4 848  =DISPP  ST=0   8
874 035A7 849  =DSPP02 ST=0   9              Clear USING flag
875 035AA 7A70       GOSUB  USINGp
```

```
876 035AE 560          GONC   DSPP05          No tUSING ?
877           *                                (R3 not intact from entry)
878 035B1 7471  DUSP30 GOSUB  out1tk
879           *
880 035B5 70D2  DSPP05 GOSUB  wrdscn
881 035B9 00           CON(2) =tCOMMA
882 035BB AFF          REL(3) DSPP05
883 035BE 00           CON(2) =tSEMIC
884 035C0 5FF          REL(3) DSPP05
885 035C3 00           CON(2) =tTAB
886 035C5 E20          REL(3) PRTP80
887 035C8 00           CON(2) 0
888           *
889 035CA 8E00         GOSUBL =EOLCKR          RESTORE PTR & CK FOR EOL
          00
890 035D0 4E1          GOC    PRTP70           FOUND AN EOL TOKEN?
891           *
892 035D3 7000         GOSUB  =RESPTR
893 035D7 74A2         GOSUB  exppar
894 035DB 870          ?ST=1  0                INVALID ?
895 035DE D3           GOYES  PRTP85
896           *
897 035E0 7DE0  PRTP20 GOSUB  COMCK1
898 035E4 4CC          GOC    DUSP30
899 035E7 B06          C=C+1  P
900 035EA 962          ?A=C   B                COMMA OR SEMI-COLON?
901 035ED 4C           GOYES  DUSP30
902           *
903 035EF 6000  PRTP70 GOTO   =RESPTR
904           *
905 035F3 879   PRTP80 ?ST=1  9                In a PRINT/DISP USING ?
906 035F6 E2           GOYES  PRTPE1            Yes => TAB not legal input
907 035F8 7000         GOSUB  =LPRNCK          Errors if no LPAREN
908 035FC 7770         GOSUB  ARRY01           DO NUMCK, THEN CK FOR RPAREN
909 03600 7F72         GOSUB  ntoken
910 03604 6BDF         GOTO   PRTP20
911           *
912           * PRINT# - ALLOW EXPRESSIONS & SINGLE COMMA DELIMITERS ONLY
913           *
914 03608 7000  PRT#P0 GOSUB  =COMCK+          Find & output comma
915 0360C 52E          GONC   PRTP70           NOT COMMA?
916           *
917 0360F 7C62  PRT#P  GOSUB  exppar
918 03613 44F          GOC    PRT#P0           Dummy Array?
919 03616 860          ?ST=0  0                VALID EXPRESSION?
920 03619 FE           GOYES  PRT#P0
921           *
922 0361B 878   PRTP85 ?ST=1  8                Implied DISP?
923 0361E 00           RTNYES
924 03620 6090         GOTO   NUMCK2           ILLEGAL EXPRESSION
925           *
926 03624 656E  PRTPE1 GOTO   FORPE1           ERR01 Error Exit
927           *
928           * Carry clr => no tUSING found
929           *       set => tUSING <parm> tSEMIC found; need to output tSEMIC
```

```
930                *
931 03628 7D52 =USINGp GOSUB  wrdscn
932 0362C 00           CON(2) =tUSING
933 0362E 800          REL(3) DSUSGP
934 03631 00           CON(2) 0
935 03633 5BB          GONC   PRTP70          (B.E.T.) Restores ptr
936                *
937 03636 859   DSUSGP ST=1   9               USING flag
938 03639 848          ST=0   8               Clear implied DISP flag
939 0363C 7B32         GOSUB  r3expp           (since about to destroy R3)
940 03640 863          ?ST=0  3               String expr ?
941 03643 61           GOYES  DUSP20
942 03645 7000         GOSUB  =D1C=R3
943 03649 134          D0=C
944 0364C 8E00         GOSUBL =LINP
          00
945 03652 431          GOC    Err03
946                *
947 03655 7A22         GOSUB  ntoken
948 03659 3100  DUSP20 LC(2)  =tSEMIC
949 0365D 962          ?A=C   B
950 03660 00           RTNYES
951 03662 6CAD         GOTO   PILP40          Pop lvl off stack; resptr
952                *
953 03666 63A1  Err03  GOTO   err03
954
```

```
                        STITLE Array Check / TAB
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) ARRYCK  -  Parses Doubly Dimensioned Array
                ** Name:    ARRY01  -  Parses Singly Dimensioned Array
                **
                ** Category:   PARUTL
                **
                ** Purpose:
                **      ARRYCK entry is useful for parsing one or two dimensional
                **      arrays.
                **
                **      ARRY01 is useful for parsing a single numeric expression
                **      followed by a closing parentheses; this could be a single
                **      dimension array parse or TAB parse.
                **
                ** Entry:
                **      D(A) = (AVMEME)
                **      D1 points at input stream
                **      D0 points into output buffer
                **      2 entry points:
                **      1) ARRYCK - D1 @ Left parentheses.
                **      2) ARRY01 - D1 past left parentheses.
                **
                ** Exit:
                **      Valid parse =>
                **       Return to caller with carry Set
                **       Subscript(s) output
                **       D1 points past the closing parentheses
                **       D0 points past the output subscript(s)
                **       ARRYCK entry:
                **          B(0) = # subscripts    (1 or 2)
                **
                **      Else Error exit
                **          Invalid or non-numeric expression
                **          No closing paren
                **
                ** Calls:      NUMCK, COMCK1
                **
                ** Uses.......
                **  Exclusive: A,B(A),C,D0,D1
                **  Inclusive: A-C,D(15-5),D0,D1,R0,R1,S0-S3,S7,S11,FUNCD0
                **
                ** Stk lvls:   5
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ---------------------------------
                **  07/06/82   JP           Modified documentation
                **
                ********************************************************************
                ********************************************************************
                *
```

```
1010            * Pointer should be past left paren
1011            *
1012 0366A 7220 =ARRYCK GOSUB  NUMC++          Step over ( & verify expression
1013 0366E 7F50         GOSUB  COMCK1          Check for comma
1014 03672 D1           B=0    A               Subscript initialization
1015 03674 5A0          GONC   ARRY02          1 subscript only
1016 03677 7220  ARRY01 GOSUB  NUMCK           Verify second subscript
1017 0367B D1           B=0    A               Subscript initialization
1018 0367D E5           B=B+1  A               2 subscripts
1019 0367F E5    ARRY02 B=B+1  A               1 subscript
1020 03681 3192 =PRENCK LCASC  \)\             Check for right paren
1021 03685 962          ?A=C   B
1022 03688 00           RTNYES
1023 0368A 8C00         GOLONG =ERR10           ) Expected
           00
```

```
1024                    STITLE NUMCK - Valid Numeric Expr Check
1025            ******************************************************************
1026            ******************************************************************
1027            **
1028            ** Name:(S) NUMCK   -  Valid Numeric Expression Check
1029            ** Name:(S) NUMC++  -  Move D1 1-Byte, Do Valid Numeric Expr Chec
1030            ** Name:    NUM+0   -  Move D1 1-Byte, Output Byte, Ck for Num Ex
1031            ** Name:    NUMK0   -  Output Byte, Check for Valid Numeric Expr
1032            **
1033            ** Category:   PARUTL
1034            **
1035            ** Purpose:
1036            **      Checks for and Outputs Valid Numeric Expression
1037            **      Error Exit if not found
1038            **
1039            ** Entry:
1040            **      D(A) = (AVMEME)
1041            **      D1 points at input stream
1042            **      D0 points into output buffer
1043            **      4 entry points:
1044            **
1045            **      NUMCK   - D1 points at optional blanks preceding
1046            **                  alleged numeric expression.
1047            **      NUMC++  - D1 is 1-byte prior to alleged numeric expr
1048            **      NUMC+0  - D1 is 1-byte prior to alleged numeric expr
1049            **                A(B) = byte to write to output buffer prior
1050            **                 to parsing the numeric expression.
1051            **      NUMCK0  - D1 points at optional blanks preceding
1052            **                 alleged numeric expression.
1053            **                A(B) = byte to write to output buffer prior
1054            **                 to parsing the numeric expression.
1055            **
1056            ** Exit:
1057            **
1058            **      Valid numeric expression parsed =>
1059            **          Return to caller with carry clear
1060            **          P=0
1061            **          Tokenized expression written to output buffer
1062            **          D0 points past the tokenization
1063            **          Register A contains the tokenization of the text
1064            **           FOLLOWING the numeric expression
1065            **          D1 points past the corresponding text
1066            **          R3(9-5) = the input pointer to the numeric expr
1067            **          R3(A)   = the pointer to the tokenized num. expr
1068            **          NUMCK entry:
1069            **           R3(A)   = D0 on entry
1070            **           R3(9-5) = D1 on entry
1071            **          NUMC++ entry:
1072            **           R3(A)   = D0 on entry
1073            **          NUMC+0 entry:
1074            **           The value in A(B) on entry was output prior to
1075            **           the tokenized numeric expression.
1076            **          NUMCK0 entry:
1077            **           R3(9-5) = D1 on entry
1078            **           The value in A(B) on entry was output prior to
```

```
1079                ** the tokenized numeric expression.
1080                **
1081                **      Error exit - Invalid or non-numeric expression
1082                **
1083                ** Calls:      r3exp+ (EXPPAR,R3=D1C), D1C=R3
1084                **
1085                ** Uses:       A-C,D(15-5), R0,R1,R3, S0-S3,S7,S11,
1086                **              FUNCD0
1087                **
1088                ** Stk lvls:   4
1089                **
1090                ** History:
1091                **
1092                **    Date      Programmer              Modification
1093                **  --------    ----------   ----------------------------------
1094                **  07/06/82    J.P.         Modified documentation
1095                **  11/11/82    S.W.         Added entry points NUMC+0 and NUMCK0
1096                **  05/12/83    S.W.         Eliminated NUMCK+ entry point
1097                **
1098                ********************************************************************
1099                ********************************************************************
1100 03690 171  =NUMC++ D1=D1+ 2
1101 03693 590         GONC   NUMCK          (B.E.T.)
1102                *
1103 03696 171  =NUMC+0 D1=D1+ 2
1104 03699 7C80 =NUMCK0 GOSUB  out1tk
1105                *
1106 0369D 76D1 =NUMCK  GOSUB  r3exp+         Save input ptr, eval expression
1107 036A1 863         ?ST=0   3              Legal string expression?
1108 036A4 90          GOYES   NUMCK1
1109 036A6 870         ?ST=1   0              Invalid expression?
1110 036A9 80          GOYES   NUMCK2
1111 036AB 03          RTNCC
1112 036AD 7000  NUMCK1 GOSUB  =D1C=R3        Restore input pointer
1113 036B1 854   NUMCK2 ST=1   4              Don't restore during Error Exit
1114 036B4 8C00  NUMCK3 GOLONG =ERR02
          00
```

```
1115                    STITLE STRGCK - String Expr Check
1116                ********************************************************************
1117                ********************************************************************
1118                **
1119                ** Name:(S) STRGCK  -  Valid String Expression Check
1120                **
1121                ** Category:   PARUTL
1122                **
1123                ** Purpose:
1124                **      Valid String Expression Check
1125                **
1126                ** Entry:
1127                **      D1 @ Start of Alleged String expresssion
1128                **      D(A) = (AVMEME)
1129                **      D0 points into output buffer
1130                **
1131                ** Exit:
1132                **     Valid string expresion =>
1133                **      Return to caller with carry clear
1134                **      P=0
1135                **      Tokenized string expression written to output buffer
1136                **      D0 past string expression tokenization
1137                **      A = tokenization of text FOLLOWING string expression
1138                **      D1 past corresponding text of tokenization in A
1139                **
1140                **     Else error exit
1141                **
1142                ** Calls:      r3exp+ (EXPPAR,R3=D10), NUMCK
1143                **
1144                ** Uses:       A-C,D(15-5),R0,R1,R3,S0-S3,S7,S11,FUNCD0,D0,D1
1145                **
1146                ** Stk lvls:   4
1147                **
1148                ** History:
1149                **
1150                **    Date      Programmer              Modification
1151                **  --------    ----------   ---------------------------------
1152                **  07/06/82    J.P.         Modified documentation
1153                **  07/06/83    S.W.         If invalid expr, don't restore ptr
1154                **
1155                ********************************************************************
1156                ********************************************************************
1157                *
1158                *
1159 036BA 79B1 =STRGCK GOSUB  r3exp+          Save D1 in R3
1160 036BE 873         ?ST=1  3               Not a STRING?
1161 036C1 40          GOYES  STRGer
1162 036C3 01          RTN
1163                *
1164 036C5 870   STRGer ?ST=1  0               Invalid expr?
1165 036C8 9E          GOYES  NUMCK2
1166 036CA 52E         GONC   NUMCK1          (B.E.T.) Legal numeric
1167                *
```

```
1168                    STITLE Comma Check  (COMCK)
1169                ********************************************************************
1170                ********************************************************************
1171                **
1172                ** Name:(S) COMCK   -  Comma Check
1173                ** Name:    COMCK1  -  Comma Check
1174                **
1175                ** Category:   PARUTL
1176                **
1177                ** Purpose:
1178                **      COMCK entry checks to see if the following
1179                **      text's tokenization is tCOMMA.
1180                **
1181                **      COMCK1 entry checks to see if A(B) contains tCOMMA.
1182                **
1183                ** Entry:
1184                **      COMCK  - D1 points at optional blanks preceding
1185                **                 text to tokenize.
1186                **      COMCK1:  A(B) = Token to Check
1187                **
1188                ** Exit:
1189                **      P=0
1190                **      Carry set   => tCOMMA found
1191                **                     A(B)=C(B)=tCOMMA
1192                **                     COMCK entry:
1193                **                      D1 past ascii comma
1194                **
1195                **      Carry clear => tCOMMA NOT found
1196                **                     C(B)=tCOMMA
1197                **                     COMCK entry:
1198                **                      A contains text's tokenization
1199                **                      D1 past corresponding text
1200                **
1201                ** Calls:      NTOKEN    - COMCK entry only
1202                **
1203                ** Uses:       A(B), C(B), P              -   COMCK1 entry
1204                **             A-C, D1, P, S0-S3,S11, R0  -   COMCK entry
1205                **
1206                ** Stk lvls:   3  -  COMCK  entry
1207                **             0  -  COMCK1 entry
1208                **
1209                ** History:
1210                **
1211                **    Date     Programmer             Modification
1212                **  --------   ----------   ---------------------------------
1213                **  07/06/82   J.P.         Modified documentation
1214                **
1215                ********************************************************************
1216                ********************************************************************
1217                *
1218 036CD 72B1 =COMCK  GOSUB  ntoken
1219 036D1 20   =COMCK1 P=     0
1220 036D3 3100        LC(2)  =tCOMMA
1221 036D7 962         ?A=C   B
1222 036DA 00          RTNYES
```

```
1223 036DC 01          RTN
```

```
1224                    STITLE OUTLIT - Output Literal
1225            ********************************************************************
1226            ********************************************************************
1227            **
1228            ** Name:(S) OUTLIT  -  Output Delimited Literal
1229            ** Name:(S) OUTLI1  -  Output Delimited Literal
1230            ** Name:    DATACK  -  Output Literal Delimited by Quotes
1231            **
1232            ** Category:   PARUTL
1233            **
1234            ** Purpose:
1235            **      OUTLIT and OUTLI1 entry points output a string of
1236            **      literals delimited by a specified delimiter (this
1237            **      delimiter may or may not be a quote).  The only
1238            **      difference between these two entry points is that OUTLIT
1239            **      takes an error exit if no closing delimiter is found;
1240            **      OUTLI1 simply returns with the carry set in this case.
1241            **
1242            **      DATACK entry parses a string delimited by either single
1243            **      or double quotes.  If no closing delimiter is found,
1244            **      DATACK takes an error exit.
1245            **
1246            ** Entry:
1247            **      D(A) = (AVMEME)
1248            **      D1 points into the input stream
1249            **      D0 points into the output buffer
1250            **      3 entry points:
1251            **      1) OUTLIT - D1 points at the delimiting character
1252            **                  A(B) contains the ascii delimiter
1253            **                  P=0
1254            **      2) DATACK - D1 points at optional blanks preceding
1255            **                  the alleged single or double quote.
1256            **      3) OUTLI1 - D1 points at 1st character after the
1257            **                  delimiter.
1258            **                  A(B) contains the ascii delimiter.
1259            **                  P=0
1260            **
1261            ** Exit:
1262            **      Carry clr =>
1263            **       D1 is advanced to the character following the
1264            **        closing delimiter.
1265            **       The literal up through the closing delimiter
1266            **        has been written to the output buffer.
1267            **       D0 points past the closing delimiter.
1268            **
1269            **      Carry set (OUTLI1 entry only) =>
1270            **       D1 is 2 nibbles past D0 (Endline)
1271            **       All characters, up to but not including D0,
1272            **        have been output
1273            **       D0 points past the characters which have been
1274            **        output
1275            **
1276            **      Else error exit (OUTLIT, DATACK only)
1277            **       w/ D1 at 0D - Error is : Quote Expected
1278            **
```

```
1279            ** Calls:      OUT1TK, GNXTCR
1280            **
1281            ** Uses.......
1282            **  Exclusive: A,B,C,S4,D0,D1
1283            **  Inclusive: A,B,C,S4,D0,D1
1284            **
1285            ** Stk lvls:   2 (OUTLI1)
1286            **             3 (OUTLIT)
1287            **
1288            ** NOTE:       It may be desirable to limit usage of OUTLIT
1289            **             to delimiters which are single or double
1290            **             quotes, since the error message generated is
1291            **             "Quote Expected".
1292            **
1293            ** History:
1294            **
1295            **    Date      Programmer             Modification
1296            **  --------    ----------   ---------------------------------
1297            **  07/06/82    J.P.         Modified documentation
1298            **  10/12/82    S.W.         OUTLI1 entry doesn't error exit
1299            **
1300            ********************************************************************
1301            ********************************************************************
1302            *
1303            *
1304 036DE 7000 =DATACK GOSUB  =GNXTCR
1305 036E2 3100         LC(2)  =a"
1306 036E6 962          ?A=C   B
1307 036E9 A0           GOYES  OUTLIT
1308 036EB 300          LC(1)  =a'
1309 036EE 966          ?A#C   B
1310 036F1 00           RTNYES
1311            *
1312 036F3 171  =OUTLIT D1=D1+ 2
1313 036F6 7F00         GOSUB  OUTLI1
1314 036FA 500          RTNNC
1315 036FD 854          ST=1   4
1316 03700 1C1          D1=D1- 2
1317 03703 8C00         GOLONG =ERR09
           00
1318            *
1319 03709 D8   =OUTLI1 B=A    A              Copy A(B)
1320 0370B 31D0         LCHEX  0D
1321 0370F 7610  OUTLI2 GOSUB  out1tk
1322 03713 14B          A=DAT1 B
1323 03716 171          D1=D1+ 2
1324 03719 960          ?A=B   B              CLOSING QUOTE?
1325 0371C D0           GOYES  out1tk
1326 0371E 962          ?A=C   B              EOL?
1327 03721 00           RTNYES
1328 03723 5BE          GONC   OUTLI2         (B.E.T.)
1329            *
1330 03726 171   out1t+ D1=D1+ 2
1331 03729 6FCC  out1tk GOTO   PILP10
1332            *
```

```
1333 0372D 14B   CHK-OD A=DAT1 B
1334 03730 31D0         LCHEX  0D
1335 03734 966          ?A#C   B
1336 03737 FE           GOYES  outlt+
1337 03739 07           C=RSTK                 Pop level off return stack
1338 0373B 5DE          GONC   outltk          (B.E.T.) Output 0D, rtn to driver
1339             *
```

```
1340                    STITLE Output Variable  (OUTVAR)
1341                ****************************************************************
1342                ****************************************************************
1343                **
1344                ** Name:(S) OUTVAR  -  Output Parsed Variable
1345                **
1346                ** Category:   PARUTL
1347                **
1348                ** Purpose:
1349                **      Writes tokenized variable in A to the output buffer
1350                **
1351                ** Entry:
1352                **      P=0
1353                **      D(A) = (AVMEME)
1354                **      DO points into output buffer
1355                **      Register A contains variable tokenization
1356                **       from NTOKEN call
1357                **
1358                ** Exit:
1359                **      Carry Clear
1360                **      Variable tokenization written to output buffer
1361                **      DO past the tokenization
1362                **
1363                ** Calls:      OUT1TK, ARANGE
1364                **
1365                ** Uses:       A, C(A), DO
1366                **
1367                ** Stk lvls:   2
1368                **
1369                ** History:
1370                **
1371                **    Date      Programmer              Modification
1372                **  --------   ----------   ----------------------------------
1373                **  07/06/82   J.P.         Modified documentation
1374                **
1375                ****************************************************************
1376                ****************************************************************
1377                *
1378                *
1379 0373E 77EF  =OUTVAR GOSUB  out1tk
1380 03742 8E00          GOSUBL =ARANGE
            00
1381 03748 500           RTNNC                   IF LETTER, THEN DONE
1382 0374B BF4           ASR    W
1383 0374E BF4           ASR    W
1384 03751 4CE           GOC    OUTVAR           (B.E.T.)
```

```
1385                    STITLE KEY Parse
1386            ********************************************************************
1387            ********************************************************************
1388            **
1389            ** Name:     KEYP     -  DEF KEY Statement Parse
1390            ** Name:     DFKEYP   -  DEF KEY Statement Parse
1391            **
1392            ** Category: STPARS
1393            **
1394            ** Purpose:  KEYP entry is used when KEY is the first keyword
1395            **           in the statement.
1396            **
1397            **           DFKEYP entry is used when the syntax starts with
1398            **           'DEF KEY'.
1399            **
1400            **           These two entry points parse statements of the
1401            **           form:
1402            **          [DEF] KEY string expr [<,string expr[<:|;>]]
1403            **
1404            ** Entry:    D(A) = (AVMEME)
1405            **           D1 points into input stream
1406            **           D0 points into output stream
1407            **           2 entry points:
1408            **           1) KEYP   - D1 past KEY keyword
1409            **                       D0 past tKEY
1410            **           2) DFKEYP - D1 past DEF KEY keywords
1411            **                       D0 past tDEF and tKEY
1412            **
1413            ** Exit:     Legal syntax =>
1414            **            Return with carry clear
1415            **            P=0
1416            **            D1 past syntactically correct statement
1417            **            Tokenized KEY statement written to output buf.
1418            **            D0 past KEY statement tokenization
1419            **
1420            **           Else error exit
1421            **
1422            ** Calls:    RESPTR, OUT1TK, STRGCK, COMCK+
1423            **
1424            ** Uses:     A-C,D(15-5),D0,D1,R0,R1,R3,S0-S3,S7,S11,FUNCD0
1425            **
1426            ** Stk lvls: 5
1427            **
1428            ** Detail:   Tokenized KEY statement format:
1429            **          tKEY <string expr> [tCOMMA <string expr> [;|:] ]
1430            **
1431            ** History:
1432            **
1433            **    Date     Programmer   Modifications
1434            **  --------   ----------   -------------
1435            **  10/21/82   S.W.         Added special header
1436            **  05/11/83   S.W.         Replaced call to COMCK1 with COMCK+
1437            **
1438            ********************************************************************
1439            ********************************************************************
```

```
1440                *
1441                *
1442 03754 183   =DFKEYP DO=DO- 4              BACK UP PTR
1443 03757 7ECF          GOSUB  out1tk         OUTPUT tKEY
1444 0375B 7B5F  =KEYP   GOSUB  STRGCK
1445 0375F 7000          GOSUB  =COMCK+        Find & output comma token
1446 03763 5D3           GONC   POKEP9         Not a comma?
1447                * FOUND COMMA TOKEN
1448 03766 705F          GOSUB  STRGCK
1449                *
1450 0376A 31A3          LCASC  \:\
1451 0376E 962           ?A=C   B
1452 03771 F0            GOYES  KEYP10
1453 03773 3100          LC(2)  =tSEMIC
1454 03777 966           ?A#C   B              NOT COLON OR SEMI-COLON?
1455 0377A 72            GOYES  POKEP9
1456 0377C 31B3          LCASC  \;\
1457 03780 6D32  KEYP10  GOTO   outbyt         WRITE OUT TOKEN
1458                *
```

```
1459                    STITLE Optional String Parse
1460                ********************************************************************
1461                ********************************************************************
1462                **
1463                ** Name:    OPSTRp  -  Optional String Parse
1464                **
1465                ** Category:   PARUTL
1466                **
1467                ** Purpose:
1468                **      Parses a statement, allowing 1 OPTIONAL string expression
1469                **
1470                ** Entry:
1471                **      D(A) = (AVMEME)
1472                **      D1 points to alleged string expression or stmt end
1473                **      D0 points into output buffer
1474                **
1475                ** Exit:
1476                **      If Statement end or valid string expression found:
1477                **        Return with carry clear
1478                **        P      =  0
1479                **        If valid string expression found:
1480                **          D1 past expression
1481                **          Tokenized expression written to output buffer
1482                **          D0 past tokenized expression
1483                **        If statement end found:
1484                **          D1/D0 are unchanged from entry, except that
1485                **          D1 is advanced beyond any leading blanks.
1486                **
1487                **      Else error exit
1488                **
1489                ** Calls:      EOLCK, RESPTR, STRGCK
1490                **
1491                ** Uses.....  A-C,D(15-5),D0,D1,R0,R1,R3,S0-S3,S7,S11,FUNCD0
1492                **
1493                **
1494                ** Stk lvls:   5
1495                **
1496                ** History:
1497                **
1498                **    Date      Programmer              Modification
1499                **  --------   ----------   ---------------------------------
1500                **  11/20/82   S.W.         Written for ENDLINE parse
1501                **
1502                ********************************************************************
1503                ********************************************************************
1504                *
1505                *
1506 03784 7F64 =OPSTRp GOSUB  eolck+          No rtn if endline found
1507 03788 541         GONC   STRNGP          If EOL not found, RESPTR already do
1508 0378B 451         GOC    resptr          B.E.T.
1509                *
1510                *
```

```
1511                    STITLE POKE Parse
1512            ********************************************************************
1513            ********************************************************************
1514            **
1515            ** Name:    POKEP   -  POKE Statement Parse
1516            ** Name:(S) STRNGP  -  Parse of a Mandatory String Expression
1517            **
1518            ** Category: STPARS
1519            **
1520            ** Purpose:  POKEP parses POKE statement.
1521            **
1522            **           STRNGP parses a mandatory string expression
1523            **
1524            ** Entry:    D(A) = (AVMEME)
1525            **           D1 points to input stream
1526            **           D0 points into output buffer
1527            **           2 entry points:
1528            **           1) POKEP  - D1 past POKE keyword
1529            **                       D0 past tPOKE
1530            **           2) STRNGP - D1 pts to alleged string expr
1531            **
1532            ** Exit:
1533            **           Valid parse =>
1534            **            Return with carry clear
1535            **            P=0
1536            **            POKEP entry:
1537            **             D1 points past syntactically correct stmt.
1538            **             POKE tokenization written to ouput buffer.
1539            **             D0 points past POKE tokenization.
1540            **            STRNGP entry:
1541            **             D1 points past string expression.
1542            **             String expr tokenization in output buffer.
1543            **             D0 points past string expr tokenization.
1544            **
1545            **           Else error exit
1546            **
1547            ** Calls:    OUT1TK, STRGCK, COMCK+
1548            **
1549            ** Uses:     A-C,D(15-5),D0,D1,R0,R1,R3,S0-S3,S7,S11,FUNCD0
1550            **
1551            ** Stk lvls: 5
1552            **
1553            ** Detail:   POKE <string expression>,<string expression>
1554            **
1555            ** History:
1556            **
1557            **    Date     Programmer   Modifications
1558            **  --------   ----------   -------------
1559            **  05/11/83   S.W.         Replaced call to COMCK1 w/COMCK+
1560            **
1561            ********************************************************************
1562            ********************************************************************
1563            *
1564            *
1565 0378E 782F =POKEP  GOSUB  STRGCK
```

```
1566 03792 7000          GOSUB  =COMCK+
1567 03796 460           GOC    STRNGP          Found & output comma ?
1568 03799 6A8E          GOTO   PRTPE1          SYNTAX ERROR
1569 0379D 791F =STRNGP  GOSUB  STRGCK
1570 037A1       resptr
1571 037A1 6000  POKEP9  GOTO   =RESPTR
```

```
1572                    STITLE Clock System Parse
1573            ****************************************************************
1574            ****************************************************************
1575            **
1576            ** Name:    CLKPRS  -  Clock System Parse Routine
1577            **
1578            ** Category:   STPARS
1579            **
1580            ** Purpose:
1581            **      Parses SETTIME, ADJUST, ADJABS, and SETDATE stmts.
1582            **
1583            **      This parse is also useful for parsing a valid
1584            **      string OR numeric expression.
1585            **
1586            ** Entry:
1587            **      D(A) = (AVMEME)
1588            **      D1 points past SETTIME, ADJUST, ADJABS, or SETDATE
1589            **       keyword
1590            **      D0 points past the corresponding token in the out-
1591            **       put buffer.
1592            **
1593            ** Exit:
1594            **      Valid expression found =>
1595            **        P=0
1596            **        Return with carry clear
1597            **        D1 points past valid string or numeric expression.
1598            **        Tokenized string or numeric expression written to
1599            **         output buffer.
1600            **        D0 points past tokenized string or numeric expr.
1601            **
1602            **      Else error exit
1603            **
1604            ** Calls:      EXPPAR.
1605            **
1606            ** Uses.......
1607            **  Inclusive: A-C,D(15-5),R0,R1,S0-S3,S7,S11,FUNCD0,D0,D1
1608            **
1609            ** Stk lvls:   4
1610            **
1611            ** Detail:
1612            **      SETTIME <numeric expr> | <string expr>
1613            **      ADJUST <numeric expr> | <string expr>
1614            **      ADJABS <numeric expr> | <string expr>
1615            **      SETDATE <numeric expr> | <string expr>
1616            **
1617            ** Algorithm:
1618            **      EXPPAR.
1619            **      If expression invalid then ERR02, else RESPTR.
1620            **
1621            ** History:
1622            **
1623            **     Date     Programmer           Modification
1624            **   --------   ----------   --------------------------------
1625            **   05/28/82   NM           Moved from clock module to here
1626            **
```

```
1627                ************************************************************
1628                ************************************************************
1629 037A5 76D0 =CLKPRS GOSUB  exppar
1630 037A9 860          ?ST=0  0          Valid expression?
1631 037AC 5F           GOYES  resptr
1632 037AE 650F         GOTO   NUMCK3
1633                *
1634                ************************************************************
1635                ************************************************************
1636                **
1637                ** Name:    SUBP    -  SUB Statement Parse
1638                **
1639                ** Category: STPARS
1640                **
1641                ** Purpose:  Parses SUB Statement
1642                **
1643                ** Entry:    D(A) = (AVMEME)
1644                **           D1 points at the input buffer past the SUB keyword.
1645                **           D0 points into the output buffer past tSUB.
1646                **
1647                ** Exit:     Valid statement parse =>
1648                **            Return with carry clear
1649                **            D1 past syntactically correct SUB statement
1650                **            SUB stmt tokenization written to output buffer.
1651                **            D0 points past the statement tokenization.
1652                **
1653                **           Else error exit
1654                **
1655                ** Calls:    SUBNMP, NTOKEN, OUT1TK, OUTVAR, GNXTCR, OUTNIB
1656                **           ARRYCK, COMCKO
1657                **
1658                ** Uses:     A-C,D(15-5),R0,R1,S0-S3,S7,S11,FUNCD0,D0,D1
1659                **
1660                ** Stk lvls: 6
1661                **
1662                ** Detail:   SUB syntax :
1663                **           SUB name  [<parm list>]
1664                **             where a parm:= #<number>|variable
1665                **       and <number> is a decimal integer 0-999.
1666                **       Parse accepts 0-999; execution accepts 0-256.
1667                **
1668                **
1669                **           SUB tokenization :
1670                **tSUB <5nib> name [tPRMST parmlist] tPRMEN [t! comment] <5nib>
1671                **
1672                **  If a remark follows the SUB statement, following tPRMEN will
1673                **  be: t! <rem stream> tEOL <5 nib field> tEOL
1674                **
1675                **       tPRMST:= tTO   &   tPRMEN:= tALL
1676                **
1677                ** History:
1678                **
1679                **    Date     Programmer   Modifications
1680                **  --------   ----------   -------------
1681                **  02/01/83   S.W.         Added documentation
```

```
1682                ** 05/03/83   S.W.        Added call to #CK
1683                ** 05/11/83   S.W.        Replaced call to COMCK w/COMCKO
1684                **
1685                ********************************************************************
1686                ********************************************************************
1687                *
1688 037B2 7480 =SUBP   GOSUB  SUBP35         ESSENTIALLY SKIPS 5 NIBS
1689 037B6 72E1        GOSUB  SUBNMP
1690 037BA 446         GOC    SUBP30
1691                *
1692 037BD 7000  SUBP10 GOSUB  =#CK
1693 037C1 5E7         GONC   SUBP45         Found # ?
1694 037C4 7BB0        GOSUB  ntoken
1695                * BETTER BE VARIABLE
1696 037C8 86B         ?ST=0  11
1697 037CB F3          GOYES  SUBPE3         ILLEGAL PARM?
1698 037CD 862         ?ST=0  2              SIMPLE VARIABLE?
1699 037D0 04          GOYES  SUBP20
1700                * ARRAY VARIABLE - REQUIRES NULL DIMENSIONS - D1 AT '('
1701 037D2 E4          A=A+1  A              DUMMY ARRAY TOKEN
1702 037D4 766F        GOSUB  =OUTVAR        OUTPUT VARIABLE NAME
1703 037D8 7000        GOSUB  =GNXCR+
1704 037DC D1          B=0    A              PARM COUNT
1705 037DE 31C2        LCASC  \,\
1706 037E2 966         ?A#C   B              1 PARM?
1707 037E5 D0          GOYES  SUBP15
1708 037E7 871         ?ST=1  1
1709 037EA 02          GOYES  err03
1710 037EC E5          B=B+1  A
1711 037EE 7000        GOSUB  =GNXCR+        STEP OFF COMMA
1712 037F2 854   SUBP15 ST=1   4             Set in case of error
1713 037F5 768E        GOSUB  ARRY02         If no closing paren, will err
1714 037F9 844         ST=0   4              B(0) incremented in above call
1715 037FC D9          C=B    A              Copy B(0)
1716 037FE 171         D1=D1+ 2              STEP OFF CLOSING PAREN
1717 03801 8E00        GOSUBL =OUTNIB        OUTPUT PARM COUNT
           00
1718 03807 5C0         GONC   SUBP25         (B.E.T.)
1719                *
1720 0380A       SUBPE3
1721 0380A 8C00  err03  GOLONG =ERR03         Invalid parm
           00
1722                *
1723 03810 7A2F  SUBP20 GOSUB  =OUTVAR
1724 03814 7000  SUBP25 GOSUB  =COMCKO        Find & output comma
1725 03818 44A         GOC    SUBP10
1726 0381B 726E        GOSUB  PRENCK         No rtn if error
1727                *
1728 0381F 3100  SUBP30 LC(2)  =tPRMEN
1729 03823 DA          A=C    A              Read into A(B)
1730 03825 7000        GOSUB  =OAGNXT
1731 03829 8E00        GOSUBL =!CK2
           00
1732 0382F 4A0         GOC    SUBP35         NO REMARK?
1733 03832 3100        LC(2)  =tEOL          NEEDED FOR DECOMPILE
```

```
1734 03836 7481 =OUTB+5 GOSUB  outbyt
1735 0383A      =OUTNB4
1736 0383A 24    SUBP35 P=     4
1737 0383C 6BB4         GOTO   outnbs          SKIP 5 NIBS
1738            *
1739            *
1740 03840 72EE  SUBP45 GOSUB  out1t+          Output #
1741 03844 7B30         GOSUB  ntoken
1742            * MUST BE 1,2, OR 3 DIGIT INTEGER
1743 03848 8F00         GOSBVL =DRANGE
           000
1744 0384F 490          GOC    SUBP55
1745            *  ONE DIGIT INTEGER
1746 03852 73DE         GOSUB  out1tk
1747 03856 5DB          GONC   SUBP25          (B.E.T.)
1748            *
1749 03859 3100  SUBP55 LC(2)  =tINT2
1750 0385D 966          ?A#C   B
1751 03860 90           GOYES  SUBP60
1752 03862 7B26         GOSUB  out2tk           OUTPUT TOK & 2 DIGITS
1753 03866 5DA          GONC   SUBP25          (B.E.T.)
1754            *
1755 03869 CE    SUBP60 C=C-1  A               3 DIGIT INTEGER TOKEN
1756 0386B 966          ?A#C   B
1757 0386E C9           GOYES  SUBPE3          Illegal Parm?
1758 03870 76CF         GOSUB  SUBP35          OUTPUT TOKEN & 3 DIGITS
1759 03874 5F9          GONC   SUBP25          (B.E.T.)
1760            *
1761 03877 7C93  r3exp+ GOSUB  gnxtcr
1762 0387B 77AC  r3expp GOSUB  R3=D10          Save D1,D0 in R3
1763 0387F 6000 =exppar GOTO   =EXPPAR
1764            *
1765 03883 8C00  ntoken GOLONG =NTOKEN
           00
1766            *
1767 03889 8C00  wrdscn GOLONG =WRDSCN
           00
1768            *
```

```
1769                    STITLE CALL Parse
1770            ********************************************************************
1771            ********************************************************************
1772            **
1773            ** Name:(S) CALLP   -  CALL Statement Parse
1774            **
1775            ** Category: STPARS
1776            **
1777            ** Purpose:  Parses CALL Statement
1778            **
1779            ** Entry:    D(A) = (AVMEME)
1780            **           D1 past CALL keyword in input stream
1781            **           D0 past tCALL in output buffer
1782            **
1783            ** Exit:     Valid Statement Parse =>
1784            **            P=0
1785            **            Return with cary clear
1786            **            D1 past syntactically correct CALL statement
1787            **            CALL stmt tokenization written to output buffer.
1788            **            D0 points past statement tokenization.
1789            **
1790            **           Else error exit
1791            **
1792            **
1793            ** Calls:    SUBNMP,OUTBYT,EXPPAR,NUMCK,CNV2UC,FSPECp
1794            **           CLRPRM,COMCK,EOLCK+,DATACK,NTOKEN,OUTVAR,SBNMP0
1795            **           NUMCK3,NUMCK1,OBFSPp,CNV2UC,PRENCK,COMCK1,R3EXPP
1796            **
1797            ** Uses:     A-C,D(15-5), S0-S3,S7,S9-S11, D1,D0, R0-R3,
1798            **           FUNCD0
1799            **
1800            ** Stk lvls: 6
1801            **
1802            ** Detail:   Compiles to:
1803            **
1804            ** tCALL <name>
1805            **  [tPRMST (parm E<0|1> parm E<0|1> ... ] tPRMEN
1806            **   [tIN <filespec>]
1807            **
1808            **          where E0 (tCREF) indicates a pass by reference
1809            **            and E1 (tCVAL) indicates a call by value.
1810            **                parm:= <#num expr|variable|expression>
1811            **
1812            **                tIN is actually tSEMIC
1813            **
1814            ** History:
1815            **
1816            **   Date      Programmer   Modification
1817            **  --------   ----------   ------------
1818            **  10/11/82   S.W.         Output E1 (tCVAL) after chnl#
1819            **  11/11/82   S.W.         Added code to trap out user-defined
1820            **                           functions
1821            **  12/09/82   S.W.         CALL w/o parms allowed from keybd
1822            **  02/11/83   J.P.         Made REDPRM straight line code.
1823            **  05/03/83   S.W.         Added call to #CK
```

```
1824                ** 05/23/83   S.W.        Channel# ALWAYS tokenized as pass
1825                **                         by reference
1826                ** 06/02/83   S.W.        Don't allow user-defined functions
1827                **                         in channel numbers
1828                **
1829                ********************************************************************
1830                ********************************************************************
1831                *
1832                *
1833 0388F 6AFB  CALPE3 GOTO   FORPE1         No parameters in prog line (Syntax)
1834                *
1835                * CALL without parameters - ensure from keyboard
1836 03893 875   CALPNL ?ST=1  5              In a program line?
1837 03896 9F           GOYES  CALPE3
1838 03898 680F         GOTO   resptr         Okay from keyboard
1839                *
1840 0389C 7753 =CALLP  GOSUB  eolck+
1841 038A0 42F          GOC    CALPNL         No parameters?
1842                * Check for quoted string 1st
1843 038A3 773E         GOSUB  DATACK
1844 038A7 5B2          GONC   CALP03         Quoted string found & output?
1845                * Check for string variable
1846 038AA 75DF         GOSUB  ntoken
1847 038AE 87B          ?ST=1  11             A variable?
1848 038B1 E0           GOYES  CALP01
1849 038B3 7AEE  CALP00 GOSUB  resptr
1850 038B7 71E0         GOSUB  SUBNMP
1851 038BB 6B10         GOTO   CALP04
1852                *
1853 038BF 861   CALP01 ?ST=0  1              Numeric variable?
1854 038C2 1F           GOYES  CALP00
1855 038C4 862          ?ST=0  2              Simple string variable?
1856 038C7 80           GOYES  CALP02
1857                * Array variable - interpret as passing parameters
1858 038C9 BF4          ASR    W
1859 038CC BF4          ASR    W              Shift off tARRAY
1860 038CF 7B6E  CALP02 GOSUB  OUTVAR         Output variable
1861                *
1862                * Check for unquoted string
1863                *
1864 038D3 70D0  CALP03 GOSUB  SBNMP0
1865 038D7 4C5   CALP04 GOC    CALPIN         No parameter list?
1866                *
1867 038DA 849   CALP05 ST=0   9
1868 038DD 7000         GOSUB  =#CK
1869 038E1 490          GOC    CALP10         Not # ?
1870 038E4 859          ST=1   9              Channel# flag
1871 038E7 7B3E         GOSUB  out1t+         Step over #; Output t#
1872                *
1873 038EB 8E00  CALP10 GOSUBL =CLRPRM        Clear PRMCNT RAM location
             00
1874 038F1 768F         GOSUB  r3expp
1875 038F5 D8           B=A    A              Copy next token into B
1876 038F7 447          GOC    CALP25         VALID DUMMY ARRAY?
1877 038FA 137          CD1EX
```

```
1878 038FD 1F00          D1=(5) =PRMCNT       Read parameter count
           000
1879 03904 15B0          A=DAT1 1
1880 03908 135           D1=C
1881 0390B 90C           ?A#0   P
1882 0390E 15            GOYES  CALPE1        User defined function?
1883 03910 870           ?ST=1  0             INVALID EXPR?
1884 03913 05            GOYES  CALPE2
1885 03915 879           ?ST=1  9             Channel# ?
1886 03918 F4            GOYES  CALP20
1887 0391A 831           ?XM=0                PASSED BY REFERENCE?
1888 0391D 06            GOYES  CALP35
1889               * Passed by value
1890 0391F 3100  CALP15  LC(2)  =tCVAL
1891 03923 7790  CALP17  GOSUB  outbyt
1892 03927 D4            A=B    A
1893 03929 74AD          GOSUB  COMCK1
1894 0392D 4CA           GOC    CALP05        Another comma ?
1895 03930 7D4D          GOSUB  PRENCK        Assume end of parm list
1896               *
1897 03934 3100  CALPIN  LC(2)  =tPRMEN
1898 03938 7280          GOSUB  outbyt
1899 0393C 7476          GOSUB  CNV2UC
1900 03940 D6            C=A    A             Read in nib 4
1901 03942 3394          LCASC  \NI\
           E4
1902 03948 8A6           ?A#C   A
1903 0394B 97            GOYES  rtncc
1904               *
1905 0394D 3100          LC(2)  =tIN
1906 03951 173           D1=D1+ 4
1907 03954 7963          GOSUB  OBFSPp        OUTBYT,FSPECp
1908 03958 500           RTNNC
1909 0395B 6D51          GOTO   PRGP80        Illegal filespec
1910               *
1911 0395F 6D4D  CALPE1  GOTO   NUMCK1        Restore D1 from R3
1912 03963 605D  CALPE2  GOTO   NUMCK3        Invalid expression
1913               *
1914 03967 873   CALP20  ?ST=1  3             Numeric ?
1915 0396A 70            GOYES  CALP30
1916 0396C 879   CALP25  ?ST=1  9             Supposed to be channel # ?
1917 0396F 0F            GOYES  CALPE1         (error if dummy array or string)
1918 03971 3100  CALP30  LC(2)  =tCREF        Pass by reference
1919 03975 6DAF          GOTO   CALP17
1920               *
1921 03979 6741  SBNME1  GOTO   ASNPE+         Illegal name (Set S4-Syntax error)
1922               *
1923 0397D 182   CALP35  D0=D0- 3
1924 03980 1523          A=DAT0 X
1925 03984 162           D0=D0+ 3
1926 03987 327A          LCHEX  1A7           A7 is substring token
           1
1927 0398C 932           ?A=C   X             Substring with 1 parm?
1928 0398F 09            GOYES  CALP15
1929 03991 B26           C=C+1  XS
```

```
1930 03994 932          ?A=C   X              Substring with 2 parms?
1931 03997 88           GOYES  CALP15
1932 03999 57D          GONC   CALP30         (B.E.T.)
1933            *
1934            *
1935            ********************************************************************
1936            ********************************************************************
1937            **
1938            ** Name:    SUBNMP  -  Subprogram Name Parse
1939            ** Name:    SBNMPO  -  Checks for Start of Parameter List
1940            **
1941            ** Category: PARUTL
1942            **
1943            ** Purpose:  Parses subprogram name and checks for start of a
1944            **           parameter list.  It is used by SUB and CALL
1945            **           statement parse.
1946            **
1947            **           SUBNMP entry parses a subprogram name; it
1948            **           only allows it in the form of an unquoted string
1949            **           (Leading and trailing quotes can be checked for
1950            **           by the caller).
1951            **           Then it falls into the SBNMPO entry.
1952            **
1953            **           SBNMPO entry checks for the start of a parameter
1954            **           list, which is indicated by '('.  If found, it
1955            **           outputs tPRMST and returns with carry clear;
1956            **           otherwise it returns with carry set.
1957            **
1958            ** Entry:    D(A) = (AVMEME)
1959            **           D1 points to input stream
1960            **           D0 points into output buffer
1961            **           2 entry points:
1962            **           1) SUBNMP -  D1 at optional blanks preceding
1963            **                         the alleged subprogram name.
1964            **           2) SBNMPO -  Tokenized subprogram name output,
1965            **                         with D0 positioned past it.
1966            **                        D1 past the subprogram name.
1967            **
1968            ** Exit:     P=0
1969            **           Carry set =>
1970            **            No parameter list
1971            **            SUBNMP entry:
1972            **             D1 pts to 1st non-blank char past subp. name
1973            **             D0 points past tokenized subprogram name
1974            **            SBMNPO entry:
1975            **             D1 preserved from entry (except advanced
1976            **              past leading blanks)
1977            **             D0 preserved from entry
1978            **
1979            **           Carry clr =>
1980            **            There should be a parameter list
1981            **            D1 past ascii (
1982            **            tPRMST output; D0 points past it
1983            **            C(B)=ascii ( ; A(B)=tPRMST
1984            **
```

```
1985                ** Error exit if illegal name (SBNMP entry only)
1986                **                or if insufficient memory (both entries).
1987                **
1988                ** Calls:    FILEP, FSPC10, OUTBYT, NTOKEN, RESPTR
1989                **
1990                ** Uses:     A-C, R0, S0-S3,S7,S11
1991                **
1992                ** Stk lvls: 5
1993                **
1994                ** History:
1995                **
1996                **    Date     Programmer   Modifications
1997                **  --------   ----------   -------------
1998                **  11/04/83   S.W.         Modified documentaion header
1999                **
2000                ********************************************************************
2001                ********************************************************************
2002                *
2003 0399C 7065   SUBNMP GOSUB  FILEP-
2004 039A0 58D           GONC   SBNME1          ILLEGAL NAME?
2005 039A3 7243          GOSUB  FSPC10          Output tLITRL and name
2006 039A7 78DE   SBNMP0 GOSUB  ntoken          SCAN TO '(' OR EOL
2007 039AB 3182          LCASC  \(\
2008 039AF 962           ?A=C   B
2009 039B2 80            GOYES  SBNMP3
2010                * No parm list
2011 039B4 79ED          GOSUB  resptr
2012 039B8 02            RTNSC
2013                *
2014 039BA 3100   SBNMP3 LC(2)  =tPRMST
2015 039BE       =OUTbyt
2016 039BE 8C00   outbyt GOLONG =OUTBYT
           00
2017                *
```

```
2018                    STITLE LR Statement Parse
2019                ********************************************************************
2020                ********************************************************************
2021                **
2022                ** Name:    LRP     -  LR Statement Parse
2023                **
2024                ** Category: STPARS
2025                **
2026                ** Purpose:  Parses LR (Linear Regression) Statement
2027                **
2028                ** Entry:    D(A) = (AVMEME)
2029                **           D1 points into input buffer past LR keyword.
2030                **           D0 points into output buffer past tLR.
2031                **
2032                ** Exit:     Valid statement parse =>
2033                **            Return with carry clear
2034                **            P=0
2035                **            D1 points past syntactically correct LR stmt
2036                **            LR stmt tokenization written to output buffer
2037                **            D0 points past statement tokenization.
2038                **
2039                **           Else error exit
2040                **
2041                ** Calls:    NUMCK, COMCK+, DSTp
2042                **
2043                ** Uses:     A-C,D(15-5),R0-R3,S0-S3,S7,S11,FUNCD0,D0,D1
2044                **
2045                ** Stk lvls: 5
2046                **
2047                ** Detail:   LR Tokenization:
2048                **   tLR <num expr  num expr> [tCOMMA <var> [tCOMMA <var>]]
2049                **
2050                ** History:
2051                **
2052                **    Date     Programmer   Modifications
2053                **  --------   ----------   ------------
2054                **  01/13/83   S.W.         Updated documentation header
2055                **  05/18/83   S.W.         Replaced call to NVARP w/DSTp
2056                **                           (Variables now compiled as
2057                **                           expressions)
2058                **
2059                ********************************************************************
2060                ********************************************************************
2061                *
2062 039C4 03       rtncc  RTNCC
2063                *
2064 039C6 73DC     =LRP   GOSUB  NUMCK
2065 039CA 730D            GOSUB  COMCK1
2066 039CE 5A2             GONC   LRPE4
2067 039D1 78CC            GOSUB  NUMCK
2068                * OPTIONAL VARIABLE SPECIFIERS
2069 039D5 7000            GOSUB  =COMCK+        Find & output tCOMMA
2070 039D9 5F5             GONC   RANDP2
2071                * REQUIRED TO BE NUMERIC VARIABLE
2072 039DC 7000            GOSUB  =DSTp          Destination parse
```

```
2073 039E0 863          ?ST=0  3            String ?
2074 039E3 21           GOYES  LRPe
2075 039E5 7000         GOSUB  =COMCK+       Find & output tCOMMA
2076 039E9 5F4          GONC   RANDP2        No comma ?
2077 039EC 7000         GOSUB  =DSTp
2078 039F0 873          ?ST=1  3            Numeric?
2079 039F3 64           GOYES  RANDP2
2080              *
2081 039F5 6000  LRPe   GOTO   =REDPE5
2082              *
2083 039F9 8C00  LRPE4  GOLONG =ERR04        MISSING PARM
          00
2084 039FF 65BA  LRPE5  GOTO   FORPE5        ILLEGAL VARIABLE
2085              *
```

```
2086                    STITLE ADD Parse
2087                ********************************************************************
2088                ********************************************************************
2089                **
2090                ** Name:(S) ADDP    -  ADD Statement Parse
2091                **
2092                ** Category: STPARS
2093                **
2094                ** Purpose:  Parses ADD and DROP Statements
2095                **
2096                ** Entry:    D(A) = (AVMEME)
2097                **           D1 points at input stream past ADD or DROP keyword
2098                **           D0 points into output buffer past tADD or tDROP
2099                **
2100                ** Exit:     Valid statement parse =>
2101                **            Return with carry clear
2102                **            P=0
2103                **            D1 points past syntactically correct statement
2104                **            Tokenized statement written to output buffer
2105                **            D0 points past statement tokenization
2106                **
2107                **           Else error exit
2108                **
2109                ** Calls:    NUMS (NUMCK)
2110                **
2111                ** Uses:     A-C,D(15-5),R0,R1,R3,D0,D1,S0-S3,S7,S11,FUNCD0
2112                **
2113                ** Stk lvls: 6
2114                **
2115                ** Detail:   Syntax is:
2116                **           ADD | DROP [num expr [,num expr...]]
2117                **           Tokenization is:
2118                **           tADD | tDROP [num expr [num expr...]]
2119                **              (tCOMMA is NOT output between expressions)
2120                **
2121                ** History:
2122                **
2123                **    Date     Programmer   Modifications
2124                **  --------   ----------   -------------
2125                **  02/08/83   S.W.         No longer limits to 15 expr
2126                **  05/12/83   S.W.         Use SFLAG/CFLAG parse
2127                **
2128                ********************************************************************
2129                ********************************************************************
2130                *
2131                *
2132 03A03         =ADDP
2133 03A03 70F1  =DROPP  GOSUB  eolck+
2134 03A07 413           GOC    RANDP2          NO PARMS?
2135                *
2136 03A0A 6882          GOTO   NUMS            Parse at least 1 num expr
2137                *
```

```
2138                    STITLE STAT Parse
2139                ********************************************************************
2140                ********************************************************************
2141                **
2142                ** Name:     STATP   -  STAT Statement Parse
2143                **
2144                ** Category: STPARS
2145                **
2146                ** Purpose:  Parses STAT Statement
2147                **
2148                ** Entry:    D(A) = (AVMEME)
2149                **           D1 points at input stream past STAT keyword
2150                **           D0 points into output buffer past tSTAT
2151                **
2152                ** Exit:     Valid statement parse =>
2153                **            Return with carry clear
2154                **            P=0
2155                **            D1 past syntactically correct statement
2156                **            STAT statement tokenization written to output buf
2157                **            D0 points past statement tokenization
2158                **
2159                **           Else error exit
2160                **
2161                ** Calls:    VARP, OUT1TK, NUMCK
2162                **
2163                ** Uses:     A-C,D(15-5),D0,D1,R0,R1,R3,S0-S3,S11,FUNCD0
2164                **
2165                ** Detail:   Compiled STAT statement:
2166                **             tSTAT <variable name>
2167                **                 OR
2168                **             tSTAT tARRAY <variable name> <numer expr>
2169                **
2170                ** Stk lvls:  5
2171                **
2172                ** History:
2173                **
2174                **    Date     Programmer   Modifications
2175                **  --------   ----------   -------------
2176                **  01/14/83   S.W.         Updated documentation header
2177                **
2178                ********************************************************************
2179                ********************************************************************
2180                *
2181                *
2182 03A0E 7CFA =STATP  GOSUB  VARP
2183 03A12 5CE         GONC   LRPE5          String variable?
2184 03A15 862         ?ST=0  2              NOT ARRAY VARIABLE?
2185 03A18 CA          GOYES  rtncc
2186 03A1A 727C        GOSUB  NUMC++         STEP OFF LEFT PAREN
2187 03A1E 7F5C        GOSUB  PRENCK         No rtn if error
2188 03A22 03          RTNCC
2189                *
```

```
2190                    STITLE END Statement Parse
2191                ********************************************************************
2192                ********************************************************************
2193                **
2194                ** Name:    ENDP    -  END Statement Parse
2195                **
2196                ** Category: STPARS
2197                **
2198                ** Purpose:  Parses END, END ALL, END DEF, & END SUB statements
2199                **
2200                ** Entry:    D(A) = (AVMEME)
2201                **           D1 points at input stream past END keyword
2202                **           D0 points into output buffer past tEND
2203                **           S-R0-3 nonzero iff IF statement in progress
2204                **
2205                ** Exit:     Valid statement parse =>
2206                **            Return with carry clear
2207                **            P=0
2208                **            D1 points past syntactically correct statement
2209                **            END statement tokenization written to ouput buf
2210                **            D0 points past statement tokenization
2211                **
2212                **           Else error exit
2213                **            Either END SUB or END DEF in middle of IF
2214                **             or MEMERR
2215                **
2216                ** Calls:    WRDSCN, IFCK, OUTNBS
2217                **
2218                ** Uses:     A-C, R0-R2, D0,D1, S0-S3,S11
2219                **
2220                ** Stk lvls: 4
2221                **
2222                ** Detail:   Assumes: END, END ALL, END DEF, END SUB
2223                **           For END DEF and END SUB erases END token that has
2224                **           been output and outputs a ENDDEF or ENDSUB token
2225                **           followed by 5 nibble link field.
2226                **
2227                **           END DEF, END SUB are Illegal after THEN,ELSE
2228                **
2229                ** History:
2230                **
2231                **    Date     Programmer   Modifications
2232                **  --------   ----------   -------------
2233                **  01/14/83   S.W.         Updated header; modified instruc-
2234                **                           tion to change SUB to ENDSUB, etc
2235                **                           from A=A+1 W to A=A+1 A.
2236                **
2237                ********************************************************************
2238                ********************************************************************
2239                *
2240 03A24 716E =ENDP   GOSUB  wrdscn
2241 03A28 00          CON(2) =tDEF
2242 03A2A 310         REL(3) ENDP10         END DEF
2243 03A2D 00          CON(2) =tSUB
2244 03A2F E00         REL(3) ENDP10         END SUB
```

```
2245 03A32 00          CON(2) =tALL
2246 03A34 09F         REL(3) rtncc          END ALL, Rtn to LineP
2247 03A37 00          CON(2) 0              END
2248 03A39 676D  RANDP2 GOTO  resptr
2249           *
2250 03A3D 8E00  ENDP10 GOSUBL =IFCK         Not middle of IF ?
          00
2251 03A43 183         D0=D0- 4              Back up to erase END & DEF (SUB)
2252 03A46 E4          A=A+1  A              Increment SUB to ENDSUB or
2253           *                              DEF to ENDDEF
2254 03A48 26          P=     6              Output ENDSUB or ENDDEF token
2255 03A4A 6DA2        GOTO   outnbs         and 5 nibble link field
2256           *
```

```
2257                    STITLE CREATE Parse
2258                ********************************************************************
2259                ********************************************************************
2260                **
2261                ** Name:    CREATP  -  CREATE Statement Parse
2262                **
2263                ** Category: STPARS
2264                **
2265                ** Purpose:  Parses CREATE Statement
2266                **
2267                ** Entry:    D(A) = (AVMEME)
2268                **           D1 points into input stream past CREATE keyword
2269                **           D0 points into output buffer past tCREAT
2270                **
2271                ** Exit:     Valid statement parse =>
2272                **            Return with carry clear
2273                **            P=0
2274                **            D1 points past syntactically correct CREATE stmt.
2275                **            CREATE stmt tokenization written to output buffer
2276                **            D0 points past statement tokenization
2277                **
2278                **           Else error exit
2279                **
2280                ** Calls:    FSPECp, OUT2TK, COMCK, FASCFD, NUMCK,
2281                **           FLTYPp, NUMCK0, COMCK+, RESTPR, FIXP
2282                **
2283                **
2284                ** Uses:     A-C,P,XM,D0,D1,D(15-5),S0-S3,S7,S11,FUNCD0
2285                **
2286                ** Stk lvls: 6
2287                **
2288                ** Detail:   Syntax:
2289                **          CREATE <type> <file spec> [,<size>[,<len>]]
2290                **           Tokenization:
2291                **   tCREAT <file type> <file spec>
2292                **                      [tCOMMA num expr [tCOMMA num expr]]
2293                **
2294                ** History:
2295                **
2296                **    Date     Programmer   Modifications
2297                **  --------   ----------   -------------
2298                **  01/14/83   S.W.         Updated documentation header
2299                **
2300                ********************************************************************
2301                ********************************************************************
2302                *
2303                *
2304 03A4E 7F14 =CREATP GOSUB  FLTYPp          Parse & output file type
2305 03A52 7F62         GOSUB  FSPECp          PARSE FILE SPEC
2306 03A56 426          GOC    PRGP80          ILLEGAL SPECIFICATION?
2307                *
2308 03A59 707C         GOSUB  COMCK
2309 03A5D 5BD          GONC   RANDP2          NO FILE SIZE SPECIFIED?
2310 03A60 753C         GOSUB  NUMCK0          PARSE SIZE
2311 03A64 7000         GOSUB  =COMCK+         Find & output tCOMMA
```

```
2312 03A68 50D          GONC   RANDP2
2313 03A6B 491          GOC    fixp           (B.E.T.) NUMCK, RESPTR
2314            *
```

```
2315                    STITLE RANDOMIZE Parse
2316                ********************************************************************
2317                ********************************************************************
2318                **
2319                ** Name:    RANDP   -  RANDOMIZE Statement Parse
2320                **
2321                ** Category: STPARS
2322                **
2323                ** Purpose: Parses RANDOMIZE Statement
2324                **
2325                ** Entry:   D(A) = (AVMEME)
2326                **          D1 points into input stream past RANDOMIZ keyword
2327                **          D0 points into output buffer past the corr. token
2328                **          P=0
2329                **
2330                ** Exit:    Valid statement parse =>
2331                **           Return with carry clear
2332                **           D1 past syntactically correct RANDOMIZE stmt
2333                **           RANDOMIZE stmt tokenization written to output buf
2334                **           D0 past statement tokenization
2335                **
2336                **          Else error exit
2337                **
2338                ** Calls:   EOLCK, NUMCK (FIXP), CNVWUC
2339                **
2340                ** Uses:    A-C,D(15-5),S0-S3,S7,S11,R0,R1,R3,FUNCD0,D0,D1
2341                **
2342                ** Stk lvls: 6
2343                **
2344                ** Detail:  Syntax for RANDOMIZE:
2345                **            RANDOMIZE [<num expr>]
2346                **
2347                ** History:
2348                **
2349                **    Date     Programmer   Modification
2350                **  --------   ----------   ------------
2351                **  11/09/82   S.W.         Added new documentation header
2352                **  01/26/83   S.W.         IZE is no longer optional
2353                **
2354                ********************************************************************
2355                ********************************************************************
2356                *
2357                *
2358 03A6E 7645 =RANDP  GOSUB  CNVWUC          CONVERT NEXT CHAR TO UPPER CASE
2359 03A72 3154        LCASC  \E\
2360 03A76 966         ?A#C   B
2361 03A79 84          GOYES  ASNPE+
2362 03A7B 171         D1=D1+ 2               STEP OVER ASCII 'E'
2363 03A7E 7571        GOSUB  eolck+          test for end-of-line token
2364 03A82 46B         GOC    RANDP2
2365 03A85 6000 fixp   GOTO   =FIXp           NUMCK, RESPTR
2366                   *
```

```
2367                    STITLE PURGE, CAT Statement Parse
2368              ********************************************************************
2369              ********************************************************************
2370              **
2371              ** Name:     PURGEP  -  PURGE Statement Parse
2372              ** Name:     CATP    -  CAT Statement Parse
2373              **
2374              ** Category: STPARS
2375              **
2376              ** Purpose:  Parses PURGE and CAT statements
2377              **
2378              **           The PURGEP entry point accepts a file specifier,
2379              **           the ALL keyword, or the KEYS keyword
2380              **
2381              **           The CATP entry point will accepts any of the above,
2382              **           in addition to the CARD keyword.
2383              **
2384              ** Entry:    D(A) = (AVMEME)
2385              **           D1 points into input buffer
2386              **           D0 points into output buffer
2387              **           1) PURGEP - no add'l requirements
2388              **           2) CATP   - S9=0
2389              **
2390              ** Exit:     Valid statement parse =>
2391              **            Return with carry clear
2392              **            P=0
2393              **            D1 past valid file spec or keyword
2394              **            D0 points past corresponding tokenization
2395              **             in output buffer
2396              **
2397              **           Else error exit
2398              **
2399              ** Calls:    FPSECp, EOLCK, RESPTR
2400              **
2401              ** Uses:     A-C,D(15-5),S0-S3,S7,S9,S11,R0-R3,P,XM,FUNCD0
2402              **
2403              ** Stk lvls: 6
2404              **
2405              ** Detail:   'KEYS' and 'ALL' are acceptable keywords for both
2406              **           entry points  (CATP & PURGEP).
2407              **
2408              ** History:
2409              **
2410              **    Date     Programmer   Modifications
2411              **  --------   ----------   ------------
2412              **  11/09/82   S.W.         Added new documentation header
2413              **
2414              ********************************************************************
2415              ********************************************************************
2416              *
2417              *
2418 03A89 859  =PURGEP ST=1   9
2419 03A8C 7761 =CATP   GOSUB  eolck+
2420 03A90 560          GONC   PURGP+         EOL NOT FOUND?
2421              *
```

```
2422 03A93 6AE0          GOTO   NAMEP7
2423             *
2424             *
2425 03A97 7A22  PURGP+  GOSUB  FSPECp
2426 03A9B 500           RTNNC
2427             *
2428             * SPECIAL CHARACTER OR RESERVE WORD
2429 03A9E 3100          LC(2)  =tALL
2430 03AA2 962           ?A=C   B
2431 03AA5 A1            GOYES  CLRCA+
2432             *
2433 03AA7 3100          LC(2)  =tCARD
2434 03AAB 962           ?A=C   B               tCARD
2435 03AAE 60            GOYES  PRGP30
2436 03AB0 60B0          GOTO   RNMP45          Check for tKEYS
2437             * FOUND tCARD - SEE IF LEGAL SYNTAX
2438 03AB4 869   PRGP30  ?ST=0  9               CAT PARSE?
2439 03AB7 80            GOYES  CLRCA+
2440             *
2441 03AB9 8C00  PRGP80  GOLONG =ERR11          ILLEGAL FILE NAME
           00
2442             *
2443 03ABF 03    CLRCA+  RTNCC
2444             *
2445 03AC1 854   ASNPE+  ST=1   4
2446 03AC4 65C9  PRGPE1  GOTO   FORPE1         ERR01
2447             *
```

```
2448                    STITLE DELETE Parse
2449                ********************************************************************
2450                ********************************************************************
2451                **
2452                ** Name:    D'LTP   -  DELETE statement parse
2453                **
2454                ** Category: STPARS
2455                **
2456                ** Purpose:  Parses DELETE statement.
2457                **           Accepts either <line no.> [<,line no.>]
2458                **           or the ALL keyword
2459                **
2460                ** Entry:    D(A) = (AVMEME)
2461                **           D1 points at input stream
2462                **           D0 points at output buffer
2463                **
2464                ** Exit:     Valid statement parse =>
2465                **            Return with carry clear
2466                **            P=0
2467                **            D1 points past correct syntax
2468                **            Tokenization written to output buffer
2469                **            D0 points past tokenization
2470                **
2471                **           Else error exit
2472                **
2473                ** Calls:    WRDSCN, RESPTR
2474                **
2475                ** Uses:     A-C, P, D0,D1, S0-S3,S11, R0-R2
2476                **
2477                ** Stk lvls: 6
2478                **
2479                ** Detail:   Acceptable syntax:
2480                **            <line no.> [,<line no.>]
2481                **                   OR
2482                **            ALL
2483                **
2484                **           DELETE is not programmable
2485                **
2486                ** History:
2487                **
2488                **    Date      Programmer    Modifications
2489                **  --------    ----------    -------------
2490                **  11/04/83    S.W.          Updated documentation header
2491                **
2492                ********************************************************************
2493                ********************************************************************
2494                *
2495 03AC8 7DBD =D'LTP  GOSUB  wrdscn
2496 03ACC 00          CON(2) =tALL
2497 03ACE 6FE         REL(3) rtncc
2498 03AD1 00          CON(2) 0
2499                *
2500 03AD3 7ACC        GOSUB  resptr
2501 03AD7 61F0        GOTO   LSTP20         LOOK FOR LINE#
2502                *
```

```
2503                    STITLE ASSIGN Statement Parse
2504                ********************************************************************
2505                ********************************************************************
2506                **
2507                ** Name:    ASSNP   -  ASSIGN# statement parse
2508                **
2509                ** Category:  STPARS
2510                **
2511                ** Purpose:   Parses ASSIGN Statement
2512                **
2513                ** Entry:     D(A) = (AVMEME)
2514                **            D1 points at input stream past ASSIGN keyword
2515                **            D0 points into output buffer past corr. token
2516                **
2517                ** Exit:      Valid statement parse =>
2518                **             Return with carry clear
2519                **             D1 past syntactically correct ASSIGN stmt
2520                **             ASSIGN stmt tokenization written to output buf
2521                **             D0 points past statement tokenization
2522                **
2523                **            Else error exit
2524                **
2525                ** Calls:     NUMC++, OUT1TK, WRDSCN, #CK, FSPECp
2526                **
2527                ** Uses:      A-C,D(15-5), S0-S3,S7,S10,S11,D0,D1, FUNCD0
2528                **
2529                ** Stk lvls:  6
2530                **
2531                ** Detail:   Syntax:
2532                **           ASSIGN# <channel no.> TO *
2533                **           ASSIGN# <channel no.> TO <filespec>
2534                **
2535                **           Tokenization:
2536                **           tASSGN <channel no.> tTO *
2537                **           tASSGN <channel no.> tTO <filespec>
2538                **
2539                ** History:
2540                **
2541                **    Date     Programmer   Modifications
2542                **  --------   ----------   -------------
2543                **  12/06/82   S.W.         Documented code
2544                **
2545                ********************************************************************
2546                ********************************************************************
2547                *
2548                *
2549 03ADB 7000  =ASSNP  GOSUB  =#CK
2550 03ADF 41E         GOC    ASNPE+          # not found?
2551                *
2552 03AE2 7AAB        GOSUB  NUMC++
2553 03AE6 3100        LC(2)  =tTO
2554 03AEA 966         ?A#C   B
2555 03AED 7D          GOYES  PRGPE1
2556 03AEF 763C        GOSUB  out1tk
2557                *
```

```
2558            *CHECK FOR '*' FOR MF ASSIGN
2559            *
2560 03AF3 729D          GOSUB  wrdscn
2561 03AF7 00            CON(2) =t*
2562 03AF9 BCE           REL(3) rtncc
2563 03AFC 00            CON(2) 0
2564            *
2565 03AFE 7F9C          GOSUB  resptr
2566 03B02 7FB1          GOSUB  FSPECp
2567 03B06 500           RTNNC
2568 03B09 4FA           GOC    PRGP80          (B.E.T.) - Illegal name
2569            *
2570            *
```

```
2571                    STITLE RENAME, COPY Parse
2572            ********************************************************************
2573            ********************************************************************
2574            **
2575            ** Name:    RENAMP  -  RENAME statement parse
2576            ** Name:    COPYP   -  COPY statement parse
2577            **
2578            ** Category:   STPARS
2579            **
2580            ** Purpose:
2581            **      Parse COPY and RENAME statements
2582            **
2583            **      The differences between these two entry
2584            **      points:
2585            **      RENAMP doesn't allow the CARD keyword.
2586            **      RENAMP requires a destination file specifier
2587            **      (ie it requires the TO keyword and what follows).
2588            **
2589            ** Entry:
2590            **      D(A) = (AVMEME)
2591            **      D1 points at input buffer
2592            **      D0 points into output buffer
2593            **      S8=0
2594            **      2 entry points:
2595            **      1) COPYP  - No add'l requirements
2596            **      2) RENAMP - S9=0
2597            **
2598            ** Exit:
2599            **      Valid statement parse =>
2600            **       Return with carry clear
2601            **       P=0
2602            **       D1 past accepted syntax
2603            **       Tokenization written to output buffer
2604            **       D0 points past tokenization.
2605            **
2606            **      Else error exits:
2607            **         "Illegal Name", "Syntax", or MEMERR
2608            **
2609            ** Calls:      FSPECp, WRDSCN, EOLCK
2610            **
2611            ** Uses:  A-C,D(15-5),P,XM,D0,D1,S0-S3,S7,S8,S10,S11,
2612            **             FUNCD0,R0-R3
2613            **
2614            ** Stk lvls:   6
2615            **
2616            ** NOTE:       S8 must remain intact through all calls.
2617            **
2618            ** Detail:
2619            **
2620            **      Acceptable syntax for COPY:
2621            **      COPY [ CARD | KEYS | <file spec> ] [ TO
2622            **             CARD | KEYS | <file spec> ]
2623            **
2624            **      Acceptable syntax for RENAME:
2625            **    RENAME [KEYS|<file spec>] TO  KEYS|<file spec>
```

```
2626                **
2627                **
2628                **            S8 is the destination parse flag.
2629                **
2630                ** Algorithm:
2631                **
2632                ** COPYP2: Set Destination Parse flag      (S8)
2633                **  COPYP: Try <file specification> Parse  (FSPCdp)
2634                **         If not accepted                 (Carry set)
2635                **            If reserved word             (S7=1)
2636                **               If TO
2637                **                  If Source Parse        (S8=0)
2638                **                     go Parse Dest       (COPYP2)
2639                **               If CARD | KEYS
2640                **                     go Parse Dest       (COPYP2)
2641                **               else
2642                **                 Error Exit ---> "Illegal Name"
2643                **            else
2644                **               If Source Parse          (S8=0)
2645                **                  Set No Restore of Input Pointer flag
2646                **                  Error Exit ---> "Illegal Name"
2647                **         else
2648                **            If Destination NOT yet parsed (S8=0)
2649                **               If next token = TO         (WRDSCN)
2650                **                  go Parse Destination      (COPYP2)
2651                **               else;
2652                **            RTNCC
2653                **
2654                ** History:
2655                **
2656                **    Date     Programmer            Modification
2657                **  --------   ----------   ---------------------------------
2658                **  07/06/82   JP           Modified documentation
2659                **  10/19/82   JP           Removed PCRD as acceptable file spec
2660                **  11/29/82   S.W.         Combined RENAME/COPY parse
2661                **
2662                ********************************************************************
2663                ********************************************************************
2664                *
2665 03B0C 77E0 =COPYP  GOSUB  eolck+          Allow COPY
2666 03B10 4D6         GOC    NAMEP7          Endline?
2667 03B13 859         ST=1   9               COPY flag
2668 03B16 550         GONC   RENAMP          (B.E.T.)
2669                *
2670 03B19 858   RNMP05 ST=1   8
2671 03B1C 75A1 =RENAMP GOSUB  FSPECp
2672 03B20 572         GONC   RNMP25          LEGAL FILE NAME?
2673                * ILLEGAL FILE SPEC
2674                *
2675 03B23 867         ?ST=0  7               NOT RESERVE WORD?
2676 03B26 39          GOYES  PRGP80
2677 03B28 878         ?ST=1  8               on Destination parse?
2678 03B2B B0          GOYES  RNMP15
2679                * CHECK FOR tKEYS, tTO
2680 03B2D 3100        LC(2)  =tTO
```

```
2681 03B31 962          ?A=C   B
2682 03B34 5E           GOYES  RNMP05          tTO already output
2683               *
2684 03B36 869   RNMP15 ?ST=0  9               RENAME parse?
2685 03B39 B0           GOYES  RNMP20          Yes=> Don't check for CARD
2686 03B3B 3100         LC(2)  =tCARD
2687 03B3F 962          ?A=C   B
2688 03B42 60           GOYES  RNMP25
2689               *
2690 03B44 7910  RNMP20 GOSUB  RNMP45          If not tKEYS then error
2691               *
2692 03B48 878   RNMP25 ?ST=1  8
2693 03B4B 07           GOYES  LSTPDN
2694 03B4D 783D         GOSUB  wrdscn
2695 03B51 00           CON(2) =tTO            TO TOKEN
2696 03B53 6CF          REL(3) RNMP05
2697 03B56 00           CON(2) 0
2698               *
2699 03B58 879          ?ST=1  9               COPY Parse?
2700 03B5B 32           GOYES  NAMEP7          TO not necessary
2701               *
2702 03B5D 666F         GOTO   PRGPE1
2703               *
2704               * ONLY ALLOW RESERVE WORD tKEYS
2705               *
2706 03B61 3100  RNMP45 LC(2)  =tKEYS
2707 03B65 962          ?A=C   B
2708 03B68 35           GOYES  LSTPDN
2709 03B6A 6E4F  RNMP65 GOTO   PRGP80          ILLEGAL FILE NAME
2710               *
```

```
2711                    STITLE NAME Parse
2712                ********************************************************************
2713                ********************************************************************
2714                **
2715                ** Name:    NAMEP   -  NAME Statement Parse
2716                **
2717                ** Category: STPARS
2718                **
2719                ** Purpose:  Parses NAME statement.
2720                **
2721                **           NAMEP allows a 'simple' file name only, ie it
2722                **           doesn't allow a device specifier of any kind
2723                **           (unless of course it's in a string expression)
2724                **
2725                ** Entry:    D(A) = (AVMEME)
2726                **           D1 points at alleged file specifier in input stream
2727                **           D0 points into output buffer
2728                **
2729                ** Exit:     Valid simple file name =>
2730                **            Return with carry clear
2731                **            D1 points past file name
2732                **            Tokenized file name written to output buffer
2733                **            D0 points past tokenized file name
2734                **
2735                **           Else error exit
2736                **
2737                ** Calls:    FILEP
2738                **
2739                ** Uses:     A-C,D(15-5),R0-R3,D0,D1,S0-S3,S7,S10,S11,FUNCD0
2740                **
2741                ** Stk lvls: 5
2742                **
2743                ** Detail:   Syntax:
2744                **           NAME <file name>
2745                **         where <file name> may be an unquoted string, or a
2746                **         string expression
2747                **
2748                **           Tokenization:
2749                **           tNAME tLITRL <ascii file name>
2750                **             OR
2751                **           tNAME <string expression>
2752                **
2753                ** History:
2754                **
2755                **    Date     Programmer   Modifications
2756                ** --------   ----------   -------------
2757                ** 01/14/83   S.W.         Updated documentation header
2758                **
2759                ********************************************************************
2760                ********************************************************************
2761                *
2762 03B6E 7A23 =NAMEP  GOSUB  FILEP
2763 03B72 57F         GONC   RNMP65         ILLEGAL FILE NAME
2764                * LEGAL FILE NAME
2765 03B75 877         ?ST=1  7
```

```
2766 03B78 60          GOYES  NAMEP7         STRING EXPR?
2767 03B7A 6E61        GOTO   FSPC10         OUTPUT tLITRL BEFORE FILE NAME
2768           *
2769 03B7E 622C NAMEP7 GOTO   resptr
2770           *
```

```
2771                    $TITLE LIST, PRIVATE, MERGE, EDIT, SECURE Parse
2772            ********************************************************************
2773            ********************************************************************
2774            **
2775            ** Name:     LISTP   -  LIST Statement Parse
2776            ** Name:     PRIVTP  -  PRIVATE Statement Parse
2777            ** Name:     MERGEP  -  MERGE Statement Parse
2778            ** Name:     EDITP   -  EDIT Statement Parse
2779            ** Name:     SECURP  -  SECURE Statement Parse
2780            **
2781            ** Category: STPARS
2782            **
2783            ** Purpose:  Parses LIST, SECURE, PRIVATE, MERGE, and EDIT
2784            **           statements.
2785            **
2786            **           LISTP entry allows one or two optional line
2787            **           numbers, preceding by an optional file
2788            **           specifier (which may be the KEYS keyword).
2789            **           [<filespec>|KEYS] [<line#>[,<line#>]]
2790            **
2791            **           SECURP entry allows an optional file
2792            **           specifier, which may be the KEYS keyword.
2793            **           [<filespec>|KEYS]
2794            **
2795            **           PRIVTP entry requires a file specifier,
2796            **           and doesn't accept any keyword.
2797            **           <filespec>
2798            **
2799            **           MERGEP entry requires a file specifier,
2800            **           followed by one or two optional line
2801            **           numbers.
2802            **           <filespec> [<line#>[,<line#>]]
2803            **
2804            **           EDIT entry allows an optional file
2805            **           specifier, and doesn't accept any keyword.
2806            **           [<filespec>]
2807            **
2808            ** Entry:    D(A) = (AVMEM)
2809            **           D1 points to current pos. in input stream
2810            **           D0 points into output buffer
2811            **           5 entry points:
2812            **           1) PRIVTP - No add'l requirements
2813            **           2) MERGEP - S9=0
2814            **           3) EDITP  - No add'l requirements
2815            **           4) SECURP - S8=0
2816            **           5) LISTP  - S8=0;S9=0
2817            **
2818            ** Exit:     Valid parse =>
2819            **            Return with carry clear
2820            **            P=0
2821            **            D1 past correct syntax in input stream
2822            **            Tokenization written to output buffer
2823            **            D0 past tokenization
2824            **
2825            **           Else error exit
```

```
2826            **
2827            ** Calls:     FSPECp,D1C=R3,OUT3TK,GNXTCR,EOLCK,NTOKNL,COMCK
2828            **            LSTP40
2829            **
2830            ** Uses:      A-C,D(15-5),P,XM,D0,D1,S0-S3,S7-S9,S11,R0-R3
2831            **            FUNCD0
2832            **
2833            ** Stk lvls: 6
2834            **
2835            ** Detail:
2836            **            S8=0 => KEYS is a valid file specifier
2837            **            S9=0 => Line numbers are appropriate
2838            **
2839            ** History:
2840            **
2841            **    Date      Programmer   Modifications
2842            **  ---------  ----------   -------------
2843            **  11/09/82   S.W.         Added new documentation header
2844            **
2845            ********************************************************************
2846            ********************************************************************
2847            *
2848            *
2849 03B82 859  =PRIVTP ST=1   9              NO LINE NUMBERS
2850            * Allows line#s
2851 03B85 858  =MERGEP ST=1   8              FILE MUST BE SPECIFIED
2852 03B88 6010         GOTO   LSTP01
2853            *
2854 03B8C 858  =EDITP  ST=1   8              OK if no file name; keys illegal
2855 03B8F 859  =SECURP ST=1   9              DON'T ALLOW LINE NOS.
2856 03B92 7160 =LISTP  GOSUB  eolck+
2857 03B96 47E          GOC    NAMEP7         EOL FOUND?
2858            *
2859 03B99 7821  LSTP01 GOSUB  FSPECp
2860 03B9D 5F1          GONC   LSTP10         LEGAL FILE NAME?
2861            * ILLEGAL FILE SPEC
2862 03BA0 878          ?ST=1  8              tKEYS illegal?
2863 03BA3 7C           GOYES  RNMP65
2864            *
2865 03BA5 3100         LC(2)  =tKEYS
2866 03BA9 962          ?A=C   B
2867 03BAC 11           GOYES  LSTP10
2868            * NO FILE SPECIFIED - MAY BE LINE NOS.
2869 03BAE 879          ?ST=1  9              No line#s allowed?
2870 03BB1 9B           GOYES  RNMP65
2871            * Either illegal 1st character or Reserved word
2872 03BB3 877          ?ST=1  7              Reserved word?
2873 03BB6 4B           GOYES  RNMP65
2874 03BB8 501          GONC   LSTP20         (B.E.T.) - maybe line#
2875            *
2876 03BBB 03    LSTPDN RTNCC
2877            *
2878 03BBD 879   LSTP10 ?ST=1  9              SECURE COMMAND?
2879 03BC0 BF           GOYES  LSTPDN
2880 03BC2 770B  LSTP12 GOSUB  COMCK
```

```
2881 03BC6 57B          GONC   NAMEP7
2882            *  S9=0
2883 03BC9 7A00  LSTP20 GOSUB  LSTP40
2884            *
2885 03BCD 7CFA         GOSUB  COMCK
2886 03BD1 5CA          GONC   NAMEP7
2887            *
2888 03BD4 859          ST=1   9
2889 03BD7 8E00  LSTP40 GOSUBL =NTOKNL
           00
2890 03BDD 3100         LC(2)  =tLINE#
2891 03BE1 966          ?A#C   B
2892 03BE4 F0           GOYES  LSTPE
2893 03BE6 3100         LC(2)  =tCOMMA
2894 03BEA AEA          A=C    B
2895 03BED 8C00         GOLONG =OUT3TK
           00
2896            *
2897 03BF3 661C  LSTPE  GOTO   err03          INVALID (MISSING PARM)
2898            *
```

```
2899                    STITLE AUTO Parse
2900            ****************************************************************
2901            ****************************************************************
2902            **
2903            ** Name:    AUTOP   -  AUTO Statement Parse
2904            **
2905            ** Category:   STPARS
2906            **
2907            ** Purpose:    Parses AUTO statement.  This statement parse
2908            **             does not allow a multi-statement line to follow.
2909            **             It traps out the occurence of a statement
2910            **             terminator of '@'.
2911            **
2912            **             It allows up to two line numbers, delimited by
2913            **             a comma.
2914            **
2915            ** Entry:      D1 points at the input buffer
2916            **             D0 points into the output buffer
2917            **             D(A) = (AVMEME)
2918            **             P=0
2919            **
2920            ** Exit:       Valid statement parse =>
2921            **              Return with carry clear
2922            **              P=0
2923            **              The statement terminator is NOT '@'
2924            **              Statement tokenization written to output buffer
2925            **              D0 points past the tokenization
2926            **              D1 points past the corresponding text in the
2927            **               input stream
2928            **
2929            **             Else error exit
2930
2931            ** Calls:      NTOKNL, eolck+, OUT1TK, LISTP20
2932            **
2933            ** Uses:       A-C, D0,D1, S0-S3,S11, R0
2934            **
2935            ** Detail:     AUTO is allowed on an unnumbered multi-statement
2936            **             line, but can't be followed by an '@'.
2937            **
2938            **             The line number/increment value is tokenized
2939            **             as follows:
2940            **             [tCOMMA<line#> [tCOMMA<line#>] ]
2941            **
2942            ** Stack lvls:  6
2943            **
2944            **    Date      Programmer   Modifications
2945            **  --------    ----------   -------------
2946            **  08/04/82    S.W.         Added documentation & modified
2947            **                            parse to recognize lowercase l
2948            **
2949            ****************************************************************
2950            ****************************************************************
2951            *
2952            *  This subroutine uses 5 levels
2953            *
```

```
2954 03BF7 8E00  eolck+ GOSUBL =EOLCK+
           00
2955 03BFD 400          RTNC
2956 03C00 564          GONC   FTKYP5          (B.E.T.) RESPTR
2957               *
2958               *
2959 03C03         =AUTOP
2960 03C03 70FF         GOSUB  eolck+
2961 03C07 463          GOC    FTKYP3          EOL FOUND?
2962               * BETTER BE A LINE#
2963 03C0A 7BBF         GOSUB  LSTP20
2964 03C0E 532          GONC   CONTP2          (B.E.T.)
2965               *
2966 03C11 8C00  RUNPE8 GOLONG =ERR08          EXCESS CHARACTERS
           00
2967               *
2968 03C17 8C00  gnxtcr GOLONG =GNXTCR
           00
2969               *
```

```
2970                    STITLE FETCH/CONT Parse
2971                ********************************************************************
2972                ********************************************************************
2973                **
2974                ** Name:      FETCHP  -  FETCH Statement Parse
2975                ** Name:      CONTP   -  CONT Statement Parse
2976                **
2977                ** Category:    STPARS
2978                **
2979                ** Purpose:     Parses FETCH & CONT Statements
2980                **
2981                **              Neither FETCHP or CONTP allow a statement to
2982                **              be concatenated after this one.
2983                **
2984                **              CONTP allows an optional line number or label.
2985                **
2986                ** Entry:       D1 points at input buffer
2987                **              D0 points into output buffer
2988                **              D(A) = (AVMEME)
2989                **
2990                ** Exit:        Valid Statement Parse=>
2991                **               Return with carry clear
2992                **               P=0
2993                **               Tokenized statement written to output buffer
2994                **               D0 points past tokenized statement
2995                **               D1 pts past corresponding text in input buffer
2996                **               '@' does not follow legally parsed statement
2997                **
2998                **              Else error exit
2999                **
3000                ** Calls:     NTOKEN, WRDSCN, STRGCK, RESPTR, LBLINP
3001                **
3002                ** Uses:      A-C,D(15-5),D0,D1,S0-S3,S7,S11,S9,S10,R0,
3003                **            R1,R3,P,FUNCD0
3004                **
3005                ** Stk lvls: 6
3006                **
3007                ** Detail:    Ensures that statement doesn't end in '@'
3008                **
3009                **            FETCH [ <lineno> | <label> ]
3010                **            CONT  [ <lineno> | <label> ]
3011                **
3012                ** History:
3013                **
3014                **    Date      Programmer    Modifications
3015                **  --------    ----------    -------------
3016                **  01/14/83    S.W.          Updated documentation header
3017                **
3018                ********************************************************************
3019                ********************************************************************
3020                *
3021 03C1D 786C =FETCHP GOSUB  wrdscn
3022 03C21 00           CON(2) =tKEY
3023 03C23 710          REL(3) FTKYP2
3024 03C26 00           CON(2) 0
```

```
3025                *
3026 03C28 757B             GOSUB  resptr
3027                *
3028 03C2C 8E00 =CONTP      GOSUBL =LBLINP        WILL RTN WHETHER FOUND OR NOT
           00
3029 03C32 7D4C  CONTP2     GOSUB  ntoken
3030 03C36 6700             GOTO   FTKYP3
3031                *
3032 03C3A 7C7A  FTKYP2     GOSUB  STRGCK         PARSE STRING EXPR
3033                *
3034 03C3E 3104  FTKYP3     LCASC  \@\
3035 03C42 962              ?A=C   B              FOLLOWED BY '@' ?
3036 03C45 CC               GOYES  RUNPE8         EXCESS CHARACTER ERROR, IF YES
3037                *
3038 03C47 695B  FTKYP5     GOTO   resptr         RETURN TO PARSE DRIVER
3039                *
```

```
3040                    STITLE RENUMBER, RETURN Parse
3041                ****************************************************************
3042                ****************************************************************
3043                **
3044                ** Name:     RENMP   -  RENUMBER Statement Parse
3045                ** Name:     RETRNp  -  RETURN Statement Parse
3046                **
3047                ** Category:   STPARS
3048                **
3049                ** Purpose:    RENMP parses the RENUMBER statement.
3050                **
3051                **             RETRNp parses a single keyword statement that
3052                **             cannot be followed by '@' when executed from
3053                **             the keyboard.  However, the statement can be
3054                **             followed by a concatenated statement when
3055                **             when on a numbered line.
3056                **
3057                ** Entry:      D1 pts to next non-blank character
3058                **             D0 points into output buffer
3059                **             D(A) = (AVMEME)
3060                **
3061                ** Exit:       Valid statement parse =>
3062                **              Return with carry clear
3063                **              P=0
3064                **              Tokenized statement written to output buffer
3065                **              D0 past the tokenization
3066                **              D1 past the corresponding text in input buf
3067                **
3068                **             Else error exit
3069                **
3070                ** Calls:    eolck+, LSTP20
3071                **
3072                ** Uses:     A-C, S0-S3,S9,S11, R0, D0,D1
3073                **
3074                ** Stk lvls: 6
3075                **
3076                ** History:
3077                **
3078                **    Date     Programmer   Modifications
3079                **  --------   ----------   -------------
3080                **  11/09/82   S.W.         Added new documentation header
3081                **
3082                ****************************************************************
3083                ****************************************************************
3084                *
3085 03C4B 78AF  =RENMP  GOSUB  eolck+
3086 03C4F 47F          GOC    FTKYP5          EOL FOUND?
3087 03C52 849          ST=0   9
3088 03C55 707F         GOSUB  LSTP20
3089 03C59 869          ?ST=0  9               1 LINE NUMBER ONLY?
3090 03C5C B0           GOYES  RENMP1
3091 03C5E 636F         GOTO   LSTP12          PARSE 3RD (& 4TH) LINE#
3092 03C62 865  =RETRNp ?ST=0  5               Keyboard entry ?
3093 03C65 DC           GOYES  CONTP2          Disallow "@"
3094 03C67 03    RENMP1 RTNCC
```

```
3095                *
```

```
3096                    STITLE RESET Parse
3097                ********************************************************************
3098                ********************************************************************
3099                **
3100                ** Name:    RESETp  -  RESET Statement Parse
3101                **
3102                ** Category:   STPARS
3103                **
3104                ** Purpose:
3105                **             Parses RESET command.
3106                **
3107                **             Allows optional CLOCK keyword.
3108                **
3109                ** Entry:      D(A) = (AVMEME)
3110                **             D1 points at input buffer
3111                **             D0 points into output buffer
3112                **
3113                ** Exit:       Valid statement parse =>
3114                **              Return with carry clear
3115                **              P=0
3116                **              Tokenized stmt written to output buffer
3117                **              D0 past the tokenization
3118                **              D1 past the corr. text in the input buffer
3119                **
3120                **             Else error exit
3121                **
3122                ** Calls:      WRDSCN
3123                **
3124                ** Uses:       A-C, R0-R2, D0,D1, S0-S3,S11
3125                **
3126                ** Stk lvls:   4
3127                **
3128                ** Detail:        Syntax:
3129                **             RESET [CLOCK]
3130                **
3131                ** History:
3132                **
3133                **    Date     Programmer              Modification
3134                **  --------   ----------   ------------------------------------
3135                **  09/21/82   NM           Wrote
3136                **
3137                ********************************************************************
3138                ********************************************************************
3139 03C69 7C1C =RESETp GOSUB  wrdscn         Look for CLOCK
3140 03C6D FE10         CON(6) =tCLOCK
            51
3141 03C73 15D          REL(3) rtncc
3142 03C76 00           NIBHEX 00
3143 03C78 572          GONC   Resptr         (B.E.T.) No CLOCK.  Done.
3144                *
```

```
3145                    STITLE SFLAG,CFLAG Parse
3146                ****************************************************************
3147                ****************************************************************
3148                **
3149                ** Name:    SFLGP   -  SFLAG, CFLAG Statement Parse
3150                **
3151                ** Category:   STPARS
3152                **
3153                ** Purpose:
3154                **      Parses the SFLAG and CFLAG statements
3155                **
3156                **      Allows either a keyword (ALL/MATH) or a list containing
3157                **      one or more numeric expressions.  See detail below.
3158                **
3159                ** Entry:      D(A) = (AVMEME)
3160                **             D1 points to ASCII input stream
3161                **             D0 points to parse token output stream
3162                **
3163                ** Exit:       Valid statement parse =>
3164                **              Return with carry clear
3165                **              P=0
3166                **              Tokenized stmt written to output buffer
3167                **              D0 past tokenized statement
3168                **              D1 past corr. text in input buffer
3169                **
3170                ** Calls:      WRDSCN, NUMCK, COMCK, RESPTR
3171                **
3172                ** Uses:       A-C,D(15-5),R0-R3,D0,D1,S0-S3,S7,S11,FUNCD0
3173                **
3174                ** Stk lvls:   5
3175                **
3176                ** Detail:
3177                **
3178                **             Syntax:
3179                **             SFLAG|CFLAG <ALL|MATH>
3180                **       SFLAG|CFLAG <num expr> [<,num expr>[<,num expr...]..]
3181                **
3182                **             Tokenization of the numeric expression list
3183                **             does not include a comma token.
3184                **
3185                ** History:
3186                **
3187                **    Date     Programmer             Modification
3188                **  --------   ----------   ---------------------------------
3189                **  06/10/82   PM           Documented routine
3190                **
3191                ****************************************************************
3192                ****************************************************************
3193 03C7B 7A0C =SFLGP  GOSUB  wrdscn         get next token, scan for match
3194 03C7F 00          CON(2) =tALL
3195 03C81 34D         REL(3) rtncc          ALL token: return with ALL outputte
3196 03C84 FE10        CON(6) =tMATH
          63
3197 03C8A A3D         REL(3) rtncc          MATH token: return with MATH output
3198 03C8D 00          CON(2) 0
```

```
3199 03C8F 7E0B          GOSUB  resptr        no match: restore pointer
3200 03C93 8E40  NUMS    GOSUBL NUMCK         at least one numeric expression
           AF
3201 03C99 743A          GOSUB  COMCK1
3202 03C9D 45F           GOC    NUMS
3203 03CA0 600B  Resptr  GOTO   resptr
3204              *
```

```
3205                    STITLE DEFAULT Parse
3206                ****************************************************************
3207                ****************************************************************
3208                **
3209                ** Name:    DEFAP   -  DEFAULT Statement Parse
3210                **
3211                ** Category:   STPARS
3212                **
3213                ** Purpose:
3214                **      Parses the DEFAULT ON | OFF | EXTEND statement
3215                **
3216                ** Entry:      D(A) = (AVMEME)
3217                **             D1 points to ascii input stream
3218                **             D0 points into token stream in output buffer
3219                **
3220                ** Exit:       Valid statement parse=>
3221                **              Return with carry clear
3222                **              P=0
3223                **              Tokenized stmt written to output buffer
3224                **              D0 pts past token stream
3225                **              D1 pts past corr. text in input stream
3226                **
3227                ** Calls:      WRDSCN
3228                **
3229                ** Uses:       A-C, R0-R2, D0,D1, S0-S3,S11
3230                **
3231                ** Stk lvls:   4
3232                **
3233                ** History:
3234                **
3235                **    Date     Programmer             Modification
3236                **  --------   ----------   ----------------------------------
3237                **  06/10/82   P.M.         Documented routine
3238                **
3239                ****************************************************************
3240                ****************************************************************
3241 03CA4 71EB =DEFAP  GOSUB  wrdscn          get next token, scan for match
3242 03CA8 00          CON(2) =tON
3243 03CAA A1D         REL(3) rtncc
3244 03CAD 00          CON(2) =tOFF
3245 03CAF 51D         REL(3) rtncc
3246 03CB2 FE10        CON(6) =tEXTND
          62
3247 03CB8 C0D         REL(3) rtncc
3248 03CBB 00          CON(2) 0
3249 03CBD 653F        GOTO   LSTPE          error: missing or illegal parm.
```

```
3250                STITLE File Specification Parse
3251            ********************************************************************
3252            ********************************************************************
3253            **
3254            ** Name:(S) FSPECp  -  File Specification Parse
3255            ** Name:    FSPC10  -  Outputs Literal File Name
3256            **
3257            ** Category:   PARUTL
3258            **
3259            ** Purpose:
3260            **      File Specficiation Parse
3261            **
3262            **      FSPECp accepts string expressions as valid file
3263            **      specifiers.  Quoted strings are considered string
3264            **      expressions.
3265            **
3266            **      Unquoted strings are carefully parsed to ensure they
3267            **      conform to the correct syntax.  File names (if they're
3268            **      given) must start with a letter and, unless a poll
3269            **      handler responds, are limited to 8 characters.
3270            **      Remaining characters may be letters or digits.
3271            **      Parse includes any device specifiers that are given.
3272            **      If a device is included, a file name is optional.
3273            **
3274            **      If a valid file name is followed by '@' or by any
3275            **      char. not in the ascii range of '.' - 'z', the file
3276            **      specifier is considered to be terminated.
3277            **
3278            **      If a valid file name is followed by ':', FSPECp
3279            **      attempts to parse the device that should follow.
3280            **      If the device is not MAIN, PORT, CARD, or PCRD,
3281            **      a device poll is done.
3282            **
3283            **      If a valid file name is followed by any other
3284            **      character (this file names over 8 characters
3285            **      long), a poll is done.
3286            **
3287            **      FSPC10 entry is used to write a legally
3288            **      parsed file name to the output buffer; it is
3289            **      generally called after FILEP has been called
3290            **      successfully.  Its entry conditions are
3291            **      matched by the exit conditions from FILEP.
3292            **
3293            ** Entry:
3294            **      D(A) = (AVMEME)
3295            **      D0 points into output buffer
3296            **      FSPECp - D1 at start of alleged file specifier
3297            **                in the input stream
3298            **      FSPC10 - File name in A
3299            **               P=0
3300            **               C(S) = #NIBS-1 in the file name
3301            **
3302            ** Exit:
3303            **   FSPECp entry
3304            **   ------------
```

```
3305            **      Carry Clear:
3306            **         P=0
3307            **         File specification accepted & output
3308            **         D0 past tokenized file spec. in output buf
3309            **         D1 past valid file specification
3310            **         S7=1 iff String expression
3311            **
3312            **      Carry Set:
3313            **         P=0
3314            **         R3(A)=D0 on entry; R3(9-5)=D1 on entry
3315            **         S7=1
3316            **            Reserved word in A
3317            **             (KEYS,ALL,TO,INTO,CARD)
3318            **            Reserved word has been output
3319            **            D0 past output reserved word in output buffer
3320            **            D1 past reserved word in input buffer
3321            **         S7=0
3322            **            Bad file parse
3323            **             (unrecognized device, extraneous chars after
3324            **              file name, invalid 1st character in file name)
3325            **            D1 restored to what it was on entry
3326            **            C(A)=D0 on entry
3327            **
3328            **      Else hard-wired error exit:
3329            **         Possibilities:
3330            **         Bad Port#          (from NUMCK)
3331            **         No closing paren   (to ERR01)
3332            **
3333            **   FSPC10
3334            **   ------
3335            **         File name written properly to output buffer:
3336            **          tLITRL <ascii file name>
3337            **         D0 points past output file name
3338            **         Carry clear
3339            **
3340            ** Calls:       FILEP,OUTNBC,POLLD+/hFSPCp,POLLD+/hDEVCp,D1C=R3
3341            **              RANGE,OUTBYT,WRDSCN,RESPTR,D=AVME,GNXTCR,OUT1TK
3342            **              NUMCK,AVS=D0
3343            **
3344            ** Uses.........
3345            **  FSPC10    : C(B), D0
3346            **  FSPECp    : A-C,P,XM,D(15-5),D0,D1,S0-S3,S7,S10,S11,R0-R3,
3347            **              FUNCD0
3348            **
3349            ** Stk lvls:   5 (FSPECp)
3350            **              2 (FSPC10)
3351            **
3352            ** Detail:      File specifiers which are unquoted strings are
3353            **              tokenized with a special 1-byte token preceding
3354            **              them:  tLITRL <unquoted string>
3355            **
3356            **              File specifiers which are string expressions
3357            **              or reserved words are NOT preceded by any such
3358            **              special byte.
3359            **
```

```
3360            **              For HPIL tokenization, see detail under
3361            **              the following poll's documentation: pFSPCp and
3362            **              pDEVCp.
3363            **
3364            ** Algorithm:
3365            **
3366            ** FSPECP: Try Mainframe File Parse (FILEP)
3367            **         If Mainframe file  (Carry set)
3368            **            If string expression (S7=1)
3369            **               Return CC
3370            **            else                  (Unquoted literal)
3371            **               If mainframe terminator
3372            **                  Output filename  (OUTNBC)
3373            **                  RTNCC
3374            **               else
3375            **      1:          If current char  = ":"
3376            **                     If filename specified
3377            **                        Output filename          (FSPC10)
3378            **                     If Mainframe Device word
3379            **                        Output Device word
3380            **                        If PORT
3381            **                           If "(" follows
3382            **                              Verify Port#       (NUMCK)
3383            **                              Verify ")"
3384            **                        RTNCC
3385            **                     else
3386            **                        Restore Input Pointer (RESPTR)
3387            **                        POLL for Device Parse
3388            **                        Return with carry as set
3389            **                  else
3390            **                     Restore D1  (R3)
3391            **                     POLL for <file spec> Parse
3392            **                     RTN with carry as returned
3393            **
3394            ** History:
3395            **
3396            **    Date     Programmer             Modification
3397            **  --------   ----------   ---------------------------------
3398            **  07/06/82   J.P.         Modified documentation
3399            **  04/08/83   S.W.         If Invalid Filespec but not reserved
3400            **                           word (on exit carry set & S7=0), the
3401            **                           D1 restored to what it was on entry
3402            **                           before return to calling routine.
3403            **
3404            ********************************************************************
3405            ********************************************************************
3406            *
3407            ********************************************************************
3408            ********************************************************************
3409            **
3410            ** Name:(S) pFSPCp  -  POLL for File Specifer Parse
3411            **
3412            ** Category:   POLL
3413            **
3414            ** Type:       POLL
```

```
3415            **
3416            ** Purpose:
3417            **      POLL for File Specification Parse.
3418            **      Unquoted string is not a legal mainframe file name.
3419            **
3420            **      Either:
3421            **        a) the 1st character isn't a letter or colon
3422            **           (device specifier starting with a character
3423            **           other than a colon)
3424            **          OR
3425            **        b) Valid file name is followed by a
3426            **           "non-terminating" character, ie one in the
3427            **           ASCII range of "." to "z" (with the exception
3428            **           of ":" and "@").  The character may be a part
3429            **           of the file name (as in a file name with more
3430            **           than 8 characters or a file name that starts
3431            **           with a letter, but contains a character other
3432            **           than a letter/digit) OR it may be a delimiter
3433            **           between the file name and the device specifier.
3434            **
3435            ** Should poll be "Handled" (return with XM=0)?:
3436            **      Yes - If File specificer is recognized
3437            **
3438            ** Meaning of "Handling" Poll (what does code do if handled?):
3439            **      Parse and tokenize file specification analogous to
3440            **      the mainframe tokenization:
3441            **
3442            **      Filename over 8 characters or a file name with a
3443            **      non letter/digit character embedded in it.
3444            **      tLITRL <ascii file name>
3445            **       Ex: ABC_X     or        ABCDEFGH
3446            **
3447            **      Filespec beginning with character other than a letter
3448            **      or a colon:
3449            **      tCOLON tLITRL <ascii file specifier>
3450            **       Ex: /WAND
3451            **
3452            **      In the first case above, if the valid file name is
3453            **      immediately followed by a 'non-terminating'
3454            **      character not recognized by the responder
3455            **      (letter in the ascii range '.' to 'z' not
3456            **      including letters/digits or '@'), a poll to pDEVCp
3457            **      may be appropriate.
3458            **      tLITRL <ascii file name> tCOLON tLITRL <ascii device>
3459            **       Ex: ABC_X.DISC    or   ABCDEFGHI/DISC
3460            **
3461            ** Entry conditions for handler (registers, ST, RAM, etc.):
3462            **      S4=S10=S7=0
3463            **      B[A] = Poll number (pFSPCp)
3464            **      HEX mode.
3465            **      P=0.
3466            **      D(A) = (AVMEME)
3467            **      D1 @ Start of File specification
3468            **           (D1 points past any preceding blanks)
3469            **      D0 @ Position in Output Buffer to begin output of
```

```
3470            **          File specification
3471            **      R3(5-9)=D1 @ start of file specfication input
3472            **      R3(A)  = D0 @ start of file specifcation output
3473            **
3474            ** Normal exit conditions from handler if handled (ST, RAM,
3475            ** registers, etc.):
3476            **      Carry clear
3477            **      P=0
3478            **      S4=S7=S10=0
3479            **      HEX mode.
3480            **      XM=0.
3481            **      File specification is accepted and output @ D0
3482            **        See NOTE below
3483            **      D0 past last token of file specificaiton
3484            **      D1 past file specification in input buffer
3485            **      R3 intact from entry
3486            **
3487            ** Normal exit conditions from handler if not handled (ST, RAM,
3488            ** registers, etc.):
3489            **      Carry clear
3490            **      P=0
3491            **      HEX mode.
3492            **      S4=S7=S10=0
3493            **      XM=1.
3494            **      R3 intact from entry
3495            **
3496            ** Error exit conditions from handler (POLL only):
3497            **      P=0
3498            **      Carry set.
3499            **      HEX mode.
3500            **      S7=S10=0
3501            **      R3 intact from entry
3502            **
3503            ** Available subroutine levels: 6
3504            **      POLL handler is one level shallower than caller--
3505            **      FSPECp uses 5; therefore Handler can use 6
3506            **
3507            **
3508            ** What registers/RAM may be used if handled?:
3509            **      A-D, D0, D1
3510            **      R0,R1,R2,R4
3511            **      STMTD1, S-R0-0, S-R0-1, SCRTCH, all of function scratch
3512            **
3513            ** What registers/RAM may be used if not handled?:
3514            **      A-D, D0, D1
3515            **      R0,R1,R2,R4
3516            **      STMTD1, S-R0-0, S-R0-1, SCRTCH, all of function scratch
3517            **
3518            ** What registers/RAM may be used if error exit (POLL only)?:
3519            **      A-D, D0, D1
3520            **      R0,R1,R2,R4
3521            **      STMTD1, S-R0-0, S-R0-1, SCRTCH, all of function scratch
3522            **
3523            ** Special memory/pointer considerations (are pointers funny?):
3524            **      No.
```

```
3525            **
3526            ** Detail:
3527            **   If HPIL is plugged in, it will answer this poll.
3528            **   Therefore, any other LEX file answering this poll
3529            **   should use an anologous tokenization scheme for the file
3530            **   name/device specifier tokenization so that file specifier
3531            **   execution works properly.
3532            **
3533            **   To get control during execution, respond to pFILXQ.  This
3534            **   is a poll that is NOT answered by HPIL.  If this poll
3535            **   is not answered by another LEX file, another poll is sent
3536            **   out later which HPIL answers.
3537            **
3538            **   For more information on how HPIL tokenizes devices, see the
3539            **   Detail portion of the documentaion on pDEVCp.
3540            **
3541            ** Envisioned application(s):
3542            **      Handle external file specifiers
3543            **        A123456789
3544            **        A123456789/DISC
3545            **        A123.WAND
3546            **        AB_X.DISC
3547            **        /WAND
3548            **
3549            ** History:
3550            **
3551            **    Date     Programmer             Modification
3552            **  --------   ----------   ----------------------------------
3553            **  07/15/82   JP           Added documentation
3554            **  05/07/83   JP           Modified documentation
3555            **
3556            ********************************************************************
3557            ********************************************************************
3558            ********************************************************************
3559            ********************************************************************
3560            **
3561            ** Name:(S) pDEVCp  -  Poll for Device Specifier Parse
3562            **
3563            ** Category:   POLL
3564            **
3565            ** Type:       POLL
3566            **
3567            ** Purpose:
3568            **      POLL for unrecognized device specifier following ":".
3569            **      If a file name preceded the colon, it has already been
3570            **      written to the output buffer.
3571            **
3572            ** Should poll be "Handled" (return with XM=0)?:
3573            **      Yes if Device specifier is recognized by handler.
3574            **
3575            ** Meaning of "Handling" Poll (what does code do if handled?):
3576            **      Parse and output tokenized form of device specifier
3577            **
3578            **      See detail below
3579            **
```

```
3580            ** Entry conditions for handler (registers, ST, RAM, etc.):
3581            **      B[A] = Poll number (pDEVCp)
3582            **      HEX mode.
3583            **      P=0.
3584            **      S4=S7=S10=0
3585            **      D1 past colon in file specification
3586            **      If a filename was specified, its tokenization was
3587            **       written to the output buffer & D0 points past the
3588            **       last character of the filename
3589            **      D(A)    = (AVMEME)
3590            **      R3(A)   = D0 @ start of tokenized filespec in output
3591            **                buffer
3592            **      R3(9-5) = D1 @ start of file spec in input buffer
3593            **
3594            ** Normal exit conditions from handler if handled (ST, RAM,
3595            ** registers, etc.):
3596            **      Carry clear
3597            **      S4=S7=S10=0
3598            **      P=0
3599            **      HEX mode.
3600            **      XM=0.
3601            **      Tokenized device written to output buffer
3602            **      D0 points past the tokenization
3603            **      D1 is past the corresponding text in the input buffer
3604            **      R3 preserved from entry
3605            **
3606            ** Normal exit conditions from handler if not handled (ST, RAM,
3607            ** registers, etc.):
3608            **      Carry clear
3609            **      S4=S7=S10=0
3610            **      P=0
3611            **      HEX mode.
3612            **      XM=1.
3613            **      Tokenized device specifier written to output buffer
3614            **      D0 points past tokenization
3615            **      D1 points past device specifier in input buffer
3616            **      R3 preserved from entry
3617            **
3618            ** Error exit conditions from handler (POLL only):
3619            **      Carry set.
3620            **      HEX mode.
3621            **      P=0
3622            **      S10=0
3623            **      R3 preserved from entry
3624            **
3625            ** Available subroutine levels: 6
3626            **      FSPECp used 5 levels
3627            **
3628            ** NOTE:
3629            **   If HPIL is plugged in, it responds to this poll;
3630            **    it accepts ALL device specifiers following the colon.
3631            **   Therefore, all LEX files should tokenize device
3632            **    specifiers in the same manner so that during execution
3633            **    the filespec execution routines work properly.
3634            **
```

```
3635            **   Respond to pFILXQ to gain control at execution.  HPIL
3636            **   does not respond to pFILXQ.
3637            **
3638            **
3639            ** Detail:
3640            **   HPIL tokenizes devices as follows:
3641            **
3642            **   device word:  (:TAPE)
3643            **    tCOLON tLITRL <ascii device word>
3644            **
3645            **   accessory ID: (:%32)
3646            **    tCOLON t% <expr> [ tCOLON <expr> ] [ tSEMIC <expr> ]
3647            **
3648            **   volume label: (.LABEL1)
3649            **    tCOLON tSEMIC <literal up to 6 chars> [ tSEMIC <expr> ]
3650            **                                            Loop #
3651            **   address:      (:1)
3652            **    tCOLON <expr> [ tSEMIC <expr> ]
3653            **           (seq#)       (loop#)
3654            **
3655            **   assign word:  (:TV)
3656            **    tCOLON tLITRL <assign word> [ tSEMIC <expr> ]
3657            **
3658            **   "*"           (*)
3659            **    tCOLON t*
3660            **
3661            ** What registers/RAM may be used if handled?:
3662            **      A-D, D0, D1
3663            **      R0,R1,R2,R4
3664            **      STMTD1, S-R0-0,S-R0-1, SCRTCH
3665            **      All of function scratch
3666            **
3667            ** What registers/RAM may be used if not handled?:
3668            **      A-D, D0, D1, P
3669            **      R0,R1,R2,R4
3670            **      STMTD1, S-R0-0,S-R0-1, SCRTCH
3671            **      All of function scratch
3672            **
3673            ** What registers/RAM may be used if error exit?:
3674            **      A-D, D0, D1, P
3675            **      R0,R1,R2,R4
3676            **      STMTD1, S-R0-0,S-R0-1, SCRTCH
3677            **      All of function scratch
3678            **
3679            ** Special memory/pointer considerations (are pointers funny?):
3680            **      No
3681            **
3682            ** Envisioned application(s):
3683            **      ABC:TAPE
3684            **      :TAPE
3685            **
3686            **
3687            ** History:
3688            **
3689            **    Date     Programmer             Modification
```

```
3690                ** --------  ----------  ----------------------------------
3691                ** 07/19/82   JP          Added documenation
3692                ** 05/08/83   JP          Modified documentation
3693                **
3694                ********************************************************************
3695                ********************************************************************
3696 03CC1 79FC  OBFSPp GOSUB  outbyt
3697 03CC5 73D1 =FSPECp GOSUB  FILEP          Mainframe file parse
3698 03CC9 AE8          B=A    B
3699 03CCC 5D3          GONC   FSPC25         Not mainframe file
3700                *
3701                * Mainframe file found
3702                *    If String expression
3703                *       RTNCC
3704                *    Check for Mainframe file terminator  [@,!,space,CR]
3705                *
3706                * *** C(S) must be preserved for later Filename write
3707                *     C(S) = # characters in Filename
3708                *
3709 03CCF 877          ?ST=1  7              String expression found ?
3710 03CD2 EC           GOYES  Resptr
3711 03CD4 14B          A=DAT1 B
3712                *
3713                * TRAP OUT LETTERS/DIGITS/@/:
3714                *
3715 03CD7 33E2          LCASC  \z.\
           A7
3716 03CDD 8E00          GOSUBL =Range
           00
3717 03CE3 AEC           ABEX   B
3718 03CE6 571           GONC   FSPC15         IN LETTER/DIGIT RANGE?
3719                *
3720                * ALL ELSE INCLUDING ,/blank
3721                *
3722                * Mainframe filename with legal terminator
3723                *    Output filename
3724                *
3725 03CE9 3100 =FSPC10 LC(2)  =tLITRL        Output "Literal" token
3726 03CED 7DCC          GOSUB  outbyt
3727 03CF1 AEA           A=C    B              Restore 1st char from OUTBYT
3728 03CF4 80DF =FSPC12 P=C    15             Move # chars to P
3729 03CF8 8C00  outnbs GOLONG =OUTNBS        Output Filename
           00
3730                *
3731                * '@' is the only legal terminator in Letter/Digit Range
3732                *
3733 03CFE 3104  FSPC15 LCASC  \@\
3734 03D02 961           ?B=C   B
3735 03D05 4E            GOYES  FSPC10
3736 03D07 5A0           GONC   FSPC30         (B.E.T.)
3737                *
3738                *
3739 03D0A 877   FSPC25 ?ST=1  7              RESERVE WORD?
3740 03D0D 00            RTNYES
3741 03D0F AF0           A=0    W              No file name specified
```

```
3742 03D12 31A3  FSPC30 LCASC  \:\           Look for colon
3743 03D16 961          ?B=C   B             Device following ?
3744 03D19 01           GOYES  MNPTP
3745             *
3746             * No ":" after Filename
3747             *    Set AVMEMS @ Current DO before Poll
3748             *      Protects Poll Save Area from overriding Parse Output
3749             *    Restore Input pointer
3750             *    POLL for <file spec> Parse
3751             *    go Handle POLL return
3752             *
3753             * D1 will point to any preceding blanks
3754             *
3755 03D1B 7D40          GOSUB  d1c=r3         RESTORE D1
3756 03D1F 7A60          GOSUB  polld+
3757 03D23 40            CON(2) =pFSPCp        <file spec> Parse
3758 03D25 6730          GOTO   FSPC35         Check POLL return
3759             *
3760             * ":" follows Filename: Try Device parse
3761             *   If filename specified
3762             *      Output File name
3763             *
3764 03D29       MNPTP
3765 03D29 978           ?A=0   W              No File name ?
3766 03D2C 60            GOYES  MNPTP1
3767 03D2E 77BF          GOSUB  FSPC10         Output File name
3768             *
3769             * Look for Mainframe Device Word
3770             *
3771 03D32 171   MNPTP1  D1=D1+ 2              Step over colon
3772 03D35 705B          GOSUB  wrdscn         Look for Mainframe Device Word
3773 03D39 00            CON(2) =tMAIN
3774 03D3B 050           REL(3) MNPTCC
3775 03D3E 00            CON(2) =tPORT
3776 03D40 430           REL(3) MNPTP5
3777 03D43 00            CON(2) =tCARD
3778 03D45 640           REL(3) MNPTCC
3779 03D48 FE10          CON(6) =tPCRD
          E3
3780 03D4E D30           REL(3) MNPTCC
3781 03D51 00            CON(2) 0
3782             *
3783             * Not Mainframe Device
3784             *    Set AVMEMS @ Current DO to prevent Poll Save area overwrite
3785             *    POLL for Device Parse - file name has been output
3786             *    If Error return from POLL
3787             *       RTNC
3788             *    If Handled by Lex File
3789             *       RTNCC
3790             *    else
3791             *       RTNSC
3792             *
3793 03D53 7A4A          GOSUB  resptr         Re-position D1 past colon
3794 03D57 7230          GOSUB  polld+
3795 03D5B 10            CON(2) =pDEVCp        Poll for Device Parse
```

```
3796 03D5D 8F00  FSPC35 GOSBVL =D=AVME
           000
3797 03D64 470          GOC    FSPC37         Error return from Lex File
3798 03D67 831          ?XM=0                 Handled by Lex File ?
3799 03D6A 12           GOYES  MNPTCC
3800              * Invalid Filespec (not Reserved word)
3801              *  Restore D1 to what it was entry
3802 03D6C         d1c=r3
3803 03D6C 8E00  FSPC37 GOSUBL =D1C=R3
           00
3804 03D72 02           RTNSC                 Not handled - Error return
3805              *
3806              * PORT Device
3807              *   Look for Port# Specifier
3808              *
3809 03D74 7F9E  MNPTP5 GOSUB  gnxtcr
3810 03D78 3182         LCASC  \(\
3811 03D7C 966          ?A#C   B
3812 03D7F C0           GOYES  MNPTCC
3813 03D81 71A9         GOSUB  out1t+         Output Left paren
3814 03D85 8C00         GOLONG =FREE15        Parse expr & closing paren
           00
3815              *
3816 03D8B 03    MNPTCC RTNCC
3817              *
3818 03D8D 8E00 =polld+ GOSUBL =AVS=D0
           00
3819 03D93 8D00         GOVLNG =POLLD+
           000
3820              *
```

```
3821            USERP   STITLE USER, LC Statement Parse
3822            ********************************************************************
3823            ********************************************************************
3824            **
3825            ** Name:    USERP   -  USER,LC Statement Parse
3826            **
3827            ** Category:   STPARS
3828            **
3829            ** Purpose:
3830            **      Parses USER | LC statement
3831            **
3832            **      Allows an optional keyword (ON|OFF)
3833            **
3834            ** Entry:      D(A) = (AVMEME)
3835            **             D1 points at input stream
3836            **             D0 points into output buffer
3837            **
3838            ** Exit:       Carry clear
3839            **             P=0
3840            **             If ON|OFF found:
3841            **              tON|tOFF has been written to the output buffer
3842            **              D0 points past the token
3843            **              D1 advanced past the keyword
3844            **             If ON|OFF not found:
3845            **              D1 has been advanced past leading blanks
3846            **              D0 is unchanged
3847            **
3848            ** Calls:      WRDSCN, RESPTR
3849            **
3850            ** Uses.......
3851            **  Exclusive: A,C
3852            **  Inclusive: A,B,C,R0-R2,D0,D1,S0-S3,S11
3853            **
3854            ** Stk lvls:   4
3855            **
3856            ** Detail:
3857            **      USER [ ( ON | OFF ) ]
3858            **      LC   [ ( ON | OFF ) ]
3859            **
3860            ** History:
3861            **
3862            **    Date      Programmer             Modification
3863            **  --------    ----------   ---------------------------------
3864            **  07/06/82    JP           Modified documentation
3865            **
3866            ********************************************************************
3867            ********************************************************************
3868 03D9A      =FLIPP
3869 03D9A 7BEA =USERP  GOSUB  wrdscn
3870 03D9E 00           CON(2) =tON           USER ON
3871 03DA0 42C          REL(3) rtncc          Return CC
3872 03DA3 00           CON(2) =tOFF          USER OFF
3873 03DA5 F1C          REL(3) rtncc          Return CC
3874 03DA8 00           NIBHEX 00             No parameter
3875 03DAA 65FE         GOTO   Resptr         Restore pointer, ReturnCC
```

```
3876            *
3877            *
3878            *
```

```
3879                    STITLE RUN / CHAIN Parse
3880            ********************************************************************
3881            ********************************************************************
3882            **
3883            ** Name:    RUNP    -  RUN statement parse
3884            ** Name:    CHAINP  -  CHAIN statement parse
3885            **
3886            ** Category:   STPARS
3887            **
3888            ** Purpose:
3889            **      RUNP parses the RUN statement.  It allows an optional
3890            **      line number or label and an optional file specifier.
3891            **
3892            **      CHAINP parses the CHAIN statement.  It requires a
3893            **      file specifier.  Line numbers and labels are
3894            **      not legal syntax for this parse.
3895            **
3896            ** Entry:      D(A) = (AVMEME)
3897            **             D1 points at the input stream
3898            **             D0 points into the output buffer
3899            **             P=0
3900            **             2 entry points:
3901            **             1) RUNP   - S9=0
3902            **             2) CHAINP - No additional requirements
3903            **
3904            ** Exit:       Valid parse =>
3905            **              Return with carry clear
3906            **              P=0
3907            **              Tokenization written to output buffer
3908            **              D0 past the tokenization
3909            **              D1 past the corr. text in the input buffer
3910            **
3911            **              Else error exit:
3912            **
3913            **        Extra Chars   - "@" follows statement
3914            **        Syntax        - Not <lineno> or <label>
3915            **        Illegal Name  - Reserve word found @ <filespec>
3916            **        Illegal Name  - Bad filespec for CHAIN
3917            **
3918            ** Calls:      FSPECp,COMCK0,OUTBYT,LBLINP,GNXTCR,LINP,RESPTR
3919            **             D1C=R3, NTOKEN,OUT1TK
3920            **
3921            ** Uses:       A-C,D(15-5), D0,D1, S0-S3,S7-S10, R0-R3,
3922            **             FUNCD0
3923            **
3924            ** Stk lvls:   6
3925            **
3926            ** Detail:
3927            **      Syntax:
3928            **      RUN [ <file spec> [ , <lineno> | <label> ] ]
3929            **      RUN , <lineno> | <label>
3930            **      RUN <lineno>
3931            **
3932            **      CHAIN <file spec>
3933            **
```

```
3934            **     Tokenization:
3935            **      tRUN tRFILE <file spec> ...
3936            **           [ tCOMMA tLINE# <jump addr> <lineno>  |
3937            **           [ tCOMMA tLBLRF <string expression>   |
3938            **           [ tCOMMA tLBLRF tLITRL <ASCII Label>  ]
3939            **      tRUN tLINE# <jump addr> <lineno>
3940            **      tRUN tCOMMA tLINE# <jump addr> <lineno> |
3941            **      tRUN tCOMMA tLBLRF <string expression>  |
3942            **           tCOMMA tLBLRF tLITRL <ASCII Label>
3943            **
3944            ** Algorithm:
3945            **
3946            ** CHAINP: Set CHAIN flag                          (S9)
3947            **   RUNP: Output Run File Token                   (tRFILE)
3948            **         Try File Spec Parse                     (FSPECp)
3949            **         Set Run File Spec flag                  (sRFILE)
3950            **         If bad file spec    (Carry set)
3951            **            If CHAIN                             (S9)
3952            **               Error --> Illegal Name
3953            **            If Reserve Word  (S7=1)
3954            **               Error --> Illegal Name
3955            **            Clear Run File Spec flag
3956            **            Restore Input Pointer                (D1C=R3)
3957            **            Back up Output Pointer over tRFILE
3958            **         If CHAIN
3959            **            go check for @                       (goto 1)
3960            **         If next token = ","                     (COMCK)
3961            **            Output comma                         (OUT1TK)
3962            **            If not Label or Line# - Carry set    (LBLINP)
3963            **               Set No Restore of Input Pointer
3964            **               Error Exit --> Syntax
3965            **               else
3966            **            else
3967            **    1:         Get next token                    (NTOKEN)
3968            **    2:         If next token = @
3969            **                  Error Exit --> Excess Chars
3970            **               else
3971            **                  RTNCC
3972            **         else
3973            **            If Run File Specified                (sRFILE)
3974            **               goto 2;
3975            **            else
3976            **               Restore input pointer             (D1C=R3)
3977            **               Try Lineno Parse                  (LINP)
3978            **               If not Lineno                     (Carry set)
3979            **                  Restore pointer
3980            **                  goto 2;
3981            **               else
3982            **                  goto 1;
3983            **
3984            ** History:
3985            **
3986            **    Date     Programmer          Modification
3987            **  --------   ----------   ----------------------------------
3988            **  07/06/82   JP           Modified documentation
```

```
3989            **  10/20/82   JP          Removed CHAIN parse
3990            **  11/16/82   JP          Added CHAIN parse
3991            **  11/18/82   JP          Added option <lineno>|<label> to CHAI
3992            **  03/23/83   JP          Removed <lineno>|<label> for CHAIN
3993            **  04/08/83   S.W.        No longer call D1C=R3 to restore D1
3994            **                          when Invalid Filespec and S7=0 -
3995            **                          this is now done by FSPECp.
3996            **
3997            ********************************************************************
3998            ********************************************************************
3999 03DAE 859  =CHAINP ST=1   9             Set CHAIN flag
4000 03DB1 3100 =RUNP   LC(2)  =tRFILE       RUN File token
4001 03DB5 780F         GOSUB  OBFSPp        OUTBYT, FSPECp
4002 03DB9 858          ST=1   sRFILE        Set Run File Specified flag
4003 03DBC 521          GONC   RUNP10        Good <file Spec>
4004            *
4005            * Bad <file spec>
4006            *   If CHAIN
4007            *      Error --> Illegal Name
4008            *   If Reserve Word
4009            *      Error --> Illegal Name
4010            *   Clear Run File Specified flag
4011            *   Restore Input pointer
4012            *   Back Output pointer over tRFILE
4013            *
4014 03DBF 879          ?ST=1  9             CHAIN ?
4015 03DC2 83           GOYES  RUNPE3        Error Exit
4016 03DC4 877          ?ST=1  7             Reserve Word ?
4017 03DC7 33           GOYES  RUNPE3        Error Exit
4018 03DC9 848          ST=0   sRFILE        Clear Run File Spec flag
4019 03DCC 181          DO=DO- 2             Back over tRFILE
4020            *
4021            * If Good File Spec and CHAIN
4022            *    go Check for "@"
4023            * Look for ","
4024            *
4025 03DCF 879   RUNP10 ?ST=1  9             CHAIN ?
4026 03DD2 02           GOYES  RUNP30
4027 03DD4 8E00         GOSUBL =COMCKO       Comma Check - Calls NTOKEN
          00
4028 03DDA 432          GOC    RUNP20        Comma
4029            *
4030            * Not comma
4031            *   If Run File specified
4032            *      Look for "@"
4033            *   else
4034            *     Look for RUN <lineno>
4035            *
4036 03DDD 878          ?ST=1  sRFILE        Run file specified ?
4037 03DE0 61           GOYES  RUNP40
4038 03DE2 7ABE         GOSUB  Resptr        Restore D1 (from COMCK)
4039 03DE6 859          ST=1   9             Line# Parse only flag
4040 03DE9 8E00         GOSUBL =LINP         Look for Line#
          00
4041 03DEF 460          GOC    RUNP40        NTOKEN done by LINP
```

```
4042                *
4043                * System Commands cannot be followed by Multi-Statements
4044                *   If next token = @
4045                *      Error Exit
4046                *   else
4047                *      Restore input pointer and return
4048                *
4049 03DF2 7D8A    RUNP30 GOSUB  ntoken          Get next token
4050 03DF6 674E    RUNP40 GOTO   FTKYP3
4051                *
4052 03DFA 6EBC    RUNPE3 GOTO   PRGP80          Illegal Name
4053                *
4054                * Comma found
4055                *   Output comma
4056                *   Look for <lineno> | <label>
4057                *
4058 03DFE 8E00    RUNP20 GOSUBL =LBLINP         Look for <label> | <lineno>
           00
4059 03E04 5DE           GONC   RUNP30          Legal parm
4060 03E07 6010          GOTO   ^ERR1           Set S4; syntax error
4061                *
4062                *
```

```
4063            TRSFMP  STITLE TRANSFORM Parse
4064            ********************************************************************
4065            ********************************************************************
4066            **
4067            ** Name:     TRSFMP  -  TRANSFORM Statement Parse
4068            **
4069            ** Category:   STPARS
4070            **
4071            ** Purpose:
4072            **      Parse TRANSFORM statement:
4073            **
4074            ** Entry:      D(A) = (AVMEME)
4075            **             D1 points at the input stream
4076            **             D1 past "TRANSFOR" keyword
4077            **             D0 past the corresponding token in output buffer
4078            **             P=0
4079            **
4080            ** Exit:       Valid statement parse =>
4081            **                Return with carry clear
4082            **                Tokenization written to output buffer
4083            **                D0 points past token stream
4084            **                D1 points past corr. text in input buffer
4085            **
4086            **             Else exits to PARERR:
4087            **          "Illegal Name" (reserved word, error)
4088            **          "Unknown file type"
4089            **
4090            ** Calls:      FSPECp, WRDSCN, CNVWUC, FLTYPp
4091            **
4092            ** Uses:       A-C,D(15-5), D1,D0, R0-R3, S0-S3,S7-S11,
4093            **             FUNCD0
4094            **
4095            ** Stk lvls:   6
4096            **
4097            ** NOTE:       S8 must remain intact through all subroutine
4098            **             calls.
4099            **
4100            ** Detail:
4101            **           Syntax:
4102            **        TRANSFORM [<file spec>] INTO <file type> [<file spec>]
4103            **
4104            ** Algorithm:
4105            **
4106            ** TRFMP2: Set Destination Parse flag      (S8)
4107            **  TRFMP: Try <file specification> Parse  (FSPCdp)
4108            **         If not accepted                 (Carry set)
4109            **            If reserved word             (S7=1)
4110            **               If TO
4111            **                  If Source Parse       (S8=0)
4112            **                     go Parse Dest      (TRFMP2)
4113            **               If CARD | KEYS
4114            **                     go Parse Dest      (TRFMP2)
4115            **               else
4116            **                 Error Exit ---> "Illegal Name"
4117            **            else
```

```
4118            **              If Source Parse          (S8=0)
4119            **                 Set No Restore of Input Pointer flag
4120            **                 Error Exit ---> "Illegal Name"
4121            **         else
4122            **            If Destination NOT yet parsed (S8=0)
4123            **               If next token = TO        (WRDSCN)
4124            **                  go Parse Destination     (TRFMP2)
4125            **               else;
4126            **            RTNCC
4127            **
4128            ** History:
4129            **
4130            **    Date     Programmer             Modification
4131            **  --------   ----------   ----------------------------------
4132            **  10/20/82   FH           Wrote
4133            **
4134            ************************************************************************
4135            ************************************************************************
4136            *
4137            *  Test for "M"
4138            *
4139 03E0B 79A1 =TRSFMP GOSUB  CNVWUC         Read and convert char to upper case
4140 03E0F 31D4         LCASC  \M\
4141 03E13 962          ?C=A   B              "M" found?
4142 03E16 B1           GOYES  TFMP15
4143 03E18 854  ^ERR1   ST=1   4              Don't restore input pointer
4144 03E1B 68AC ^ERR01  GOTO   PRGPE1         "Syntax" error
4145            *
4146            *  Parse file type
4147            *
4148 03E1F 858   TFMP10 ST=1   8              Set Destination Parse flag
4149 03E22 7B40         GOSUB  FLTYPp         Parse file type
4150 03E26 7DCD         GOSUB  eolck+
4151 03E2A 590          GONC   TFMP20         If not at eol, parse file name
4152 03E2D 6379         GOTO   resptr
4153            *
4154 03E31 171   TFMP15 D1=D1+ 2              Skip over "M"
4155 03E34 7D8E  TFMP20 GOSUB  FSPECp         Parse <file specification>
4156            *
4157            * If <file spec> NOT accepted
4158            *    If Not Reserve Word   (Illegal first character)
4159            *       If Destination Parse
4160            *          RTNCC
4161            *       else
4162            *          Set No restore of Input pointer flag
4163            *          Error Exit
4164            *
4165 03E38 5F1          GONC   TFMP60         File Spec accepted
4166 03E3B 878          ?ST=1  8              Destination Parse ?
4167 03E3E CB           GOYES  RUNPE3
4168            *
4169            * If Reserve Word
4170            *    If INTO & Source Parse
4171            *       Go parse file type
4172            *    else
```

```
4173                *       Error Exit
4174                *
4175 03E40 AF6             C=A    W
4176 03E43 35FE            LC(6)  =tINTO
           10E2
4177 03E4B 976             ?A#C   W              Not INTO token ?
4178 03E4E CA              GOYES  RUNPE3
4179 03E50 868             ?ST=0  8              Not Destination Parse ?
4180 03E53 CC              GOYES  TFMP10         Go Parse Destination
4181 03E55 54A             GONC   RUNPE3         "INTO" - Illegal Dest. filename
4182                *
4183                * If Destination NOT parsed yet  (S8=0)
4184                *    If next token = TO
4185                *       go Parse Destination file
4186                *    else
4187                *       Restore input pointer  (WRDSCN does a NTOKEN)
4188                *       RTNCC
4189                *
4190 03E58 878   TFMP60 ?ST=1  8              Destination already parsed ?
4191 03E5B 41              GOYES  TFMP70         RTNCC
4192 03E5D 782A            GOSUB  wrdscn         Look for "INTO"
4193 03E61 FE10            CON(6) =tINTO
           E2
4194 03E67 8BF             REL(3) TFMP10         Go Parse Destination
4195 03E6A 00              NIBHEX 00
4196 03E6C 5EA             GONC   ^ERR01         BET: "Syntax"
4197                *
4198 03E6F 03    TFMP70 RTNCC
4199
```

```
4200                FLTYPp  STITLE Parse File Type
4201                ********************************************************************
4202                ********************************************************************
4203                **
4204                ** Name:(S) FLTYPp  -  Parse File Type
4205                **
4206                ** Category:   PARUTL
4207                **
4208                ** Purpose:
4209                **      Parse file type specifier
4210                **
4211                ** Entry:      D(A) = (AVMEME)
4212                **             D1 points into input stream at optional blanks
4213                **              preceding the alleged file type
4214                **             D0 points into output buffer
4215                **
4216                ** Exit:       Carry clear =>
4217                **              P=0
4218                **              Valid file type found
4219                **              Tokenized file type (2 bytes) written to
4220                **               output buffer
4221                **              D0 past the tokenization in the output buffer
4222                **              D1 advanced past the corr. text
4223                **
4224                **             Else error exit to PARERR with eFTYPE
4225                **
4226                ** Calls:      FASCFD, OUT2TK, GNXTCR, STLXPT
4227                **
4228                ** Uses:       A-C, R3, S10, D0,D1, P
4229                **
4230                ** Stk lvls:   4
4231                **
4232                ** History:
4233                **
4234                **    Date     Programmer             Modification
4235                **   --------   ----------   ----------------------------------
4236                **   03/15/82   FH           Designed and coded.
4237                **   11/15/82   S.W.         Hard-wired error exit
4238                **
4239                ********************************************************************
4240                ********************************************************************
4241 03E71 72AD  =FLTYPp GOSUB  gnxtcr          Skip blanks
4242 03E75 8E00          GOSUBL =STLXPT        Store LEX pointer for error report
            00
4243 03E7B 8F00          GOSBVL =FASCFD        Find ASCII type
            000
4244 03E82 4E0           GOC    out2tk         If no error, output type as token
4245 03E85 1A00          D0=(4) =eFTYPE        Set up error code
            00
4246 03E8B 8C00          GOLONG =ERRXO
            00
4247                *
4248 03E91 8C00   out2tk GOLONG =OUT2TK        Output file type
            00
4249                *
```

```
4250                    $TITLE File Name, Label Parse
4251            ****************************************************************
4252            ****************************************************************
4253            **
4254            ** Name:(S) FILEP   -  File Name Parse
4255            ** Name:(S) LABELP  -  Label Reference Parse
4256            ** Name:(S) FILEP1  -  Literal File Name Parse
4257            ** Name:(S) FILEP-  -  Subprogram Name Parse
4258            ** Name:(S) FILEP+  -  Label Declaration Parse
4259            ** Name:(S) FILEP!  -  Literal File Name Parse
4260            **
4261            ** Category:   PARUTL
4262            **
4263            ** Purpose:    Parses a file name or a label.
4264            **             Depending on the entry point, it can allow
4265            **             string expressions and unquoted strings, or
4266            **             it can be limited to unquoted strings alone.
4267            **             However, only unquoted strings are checked
4268            **             for conformance to legal file name syntax, ie
4269            **             limited to 8 characters or less of letters
4270            **             and digits, starting with a letter.
4271            **
4272            **             FILEP and LABELP allow string expressions and
4273            **             unquoted strings.  FILEP, however, checks an
4274            **             unquoted string to ensure it is not one of
4275            **             the reserved words (TO,ALL,KEYS,CARD,INTO).
4276            **             LABELP does not make this special check.
4277            **             These entry points are useful for file name
4278            **             and label reference; for example GOTO/GOSUB
4279            **             parse calls LABELP.
4280            **
4281            **             FILEP1, FILEP-, and FILEP+ are all useful
4282            **             entry points for parsing literals which must
4283            **             conform to file name standards; included in
4284            **             this category would be label declarations
4285            **             and subprogram names in SUB statements.
4286            **             These entry points do not check for file
4287            **             reserved words.
4288            **
4289            **             FILEP! is similar to FILEP+ above, except
4290            **             it can be set to allow less than eight
4291            **             characters.
4292            **
4293            ** Entry:      D1 points at input stream
4294            **             6 ENTRY POINTS:
4295            **
4296            **             1) FILEP  - D1 points to optional blanks
4297            **                         preceding file name.
4298            **                         D(A) = (AVMEME)
4299            **                         D0 points to output buffer
4300            **             2) LABELP - Same as FILEP, except S10 must be
4301            **                         set to ignore file reserve words.
4302            **             3) FILEP1 - (LEXPTR) = address to restore input
4303            **                         pointer to; points to possible blanks
4304            **                         preceding file name.
```

```
4305          **            4) FILEP- - D1 at optional blanks preceding file
4306          **                        name.
4307          **            5) FILEP+ - D1 pointing at first character in
4308          **                        the file name.
4309          **            6) FILEP! - C(S)=#characters to allow - 1.
4310          **
4311          ** Exit:
4312          **            P=0
4313          **            S10=0 (all entries except FILEP+/FILEP!)
4314          **            S7=0 (all entries except LABELP/FILEP - see below)
4315          **            CARRY SET => IF S7=1: string expr found & output
4316          **                                  NTOKEN done on following
4317          **                                  data (LABELP/FILEP only)
4318          **                         IF S7=0: File name in A. D1 past
4319          **                                  the last legal character.
4320          **                                  C(S) set for WP write.
4321          **
4322          **            CARRY CLR => IF S7=1  Reserve word found, token
4323          **                                  output & in A(B),B(B).
4324          **                                  D1 past the reserve word.
4325          **                                  (FILEP only)
4326          **                         IF S7=0: Illegal 1st character. D1
4327          **                                  pointing to the character.
4328          **
4329          **                         R3(A)=D0 @ entry; R3(9-5)=D1 @ entry
4330          **                         Use D1C=R3 to restore D1
4331          **
4332          ** Calls:    CATCHR, RESPTR, r3expp (EXPPAR,R3=D1C),
4333          **           GNXTCR, BLANKC, WRDSCN,
4334          **
4335          ** Uses:     A-C,D(15-5),S0-S3,S7,S10,S11,D0,D1,R0-R3,FUNCD0
4336          **             (FILEP/LABELP entry)
4337          **
4338          **           A,B(A),B(S),C,D(S),S1,S2,S7,S10,D1
4339          **             (FILEP1, FILEP-)
4340          **
4341          **           FILEP+ entry uses everything FILEP1 uses except S10.
4342          **           FILEP! entry uses everything FILEP+ uses except S7.
4343          **
4344          ** Stk lvls:
4345          **  FILEP, LABELP           - 4
4346          **  all other entry points  - 3
4347          **
4348          ** History:
4349          **
4350          **    Date     Programmer   Modification
4351          **  --------   ----------   ------------
4352          **  07/08/82   S.W.         Updated documentation
4353          **  07/27/82   S.W.         Now allow unquoted 'reserve
4354          **                           words' as file names, provided
4355          **                           they're followed by a colon.
4356          **  10/18/82   JP           Removed PCRD as reserve word
4357          **  11/23/82   JP           Clearing S10 on exit
4358          **
4359          ************************************************************
```

```
4360                ************************************************************
4361                *
4362                *
4363 03E97 84A      FILPEX ST=0   10            Clear S10/future Error flag
4364 03E9A 02              RTNSC
4365 03E9C 84A      =FILEP ST=0   10            NOTE RESERVE FILE WORDS
4366 03E9F 78D9     =LABELP GOSUB  r3expp       Save pointers, eval expression
4367 03EA3 857             ST=1   7             STRING EXPRESSION FLAG
4368 03EA6 863             ?ST=0  3             VALID STRING EXPR?
4369 03EA9 EE              GOYES  FILPEX        Clear S10, Return SC
4370                * RESTORE POINTERS - BETTER BE FILE NAME WITHOUT QUOTES
4371 03EAB 7DBE            GOSUB  d1c=r3        RESTORE INPUT & OUTPUT PTRS
4372 03EAF 134             DO=C
4373                *
4374 03EB2 87A             ?ST=1  10            IGNORE POSSIBLE RESERVE WORDS?
4375 03EB5 B4              GOYES  FILEP-
4376                *
4377 03EB7 7EC9            GOSUB  wrdscn
4378                * IF ANY ENTRY IN THIS TABLE REACHES 8 CHARACTERS - PROBLEMS!!
4379 03EBB 00              CON(2) =tTO
4380 03EBD 020             REL(3) FILEP^
4381 03EC0 00              CON(2) =tALL
4382 03EC2 B10             REL(3) FILEP^
4383 03EC5 00              CON(2) =tKEYS
4384 03EC7 610             REL(3) FILEP^
4385 03ECA 00              CON(2) =tCARD
4386 03ECC 110             REL(3) FILEP^
4387 03ECF FE10            CON(6) =tINTO
             E2
4388 03ED5 800             REL(3) FILEP^
4389 03ED8 00              CON(2) 0
4390                *
4391 03EDA 512             GONC   FILEP1        (B.E.T.)
4392                *
4393                * FOUND A RESERVE WORD - SEE IF ONLY IMBEDDED IN LARGER FILE NAME
4394 03EDD AE8     FILEP^ B=A    B
4395 03EE0 14B             A=DAT1 B
4396 03EE3 31A3            LCASC  \:\
4397 03EE7 962             ?A=C   B
4398 03EEA C0              GOYES  FILEPO
4399 03EEC 7970            GOSUB  CATCH+
4400 03EF0 AE4             A=B    B
4401 03EF3 500             RTNNC                RESERVE WORD? (YES => ALREADY OUTPU
4402                * RESERVE WORD FOUND IMBEDDED IN LARGER ASCII STREAM - BACK UP
4403                * OUTPUT POINTER
4404 03EF6 11B     FILEPO C=R3
4405 03EF9 134             DO=C
4406                *
4407 03EFC 70AD    =FILEP1 GOSUB  Resptr
4408                *
4409 03F00 84A     =FILEP- ST=0   10            Clear Reserve Word/Future Error fla
4410 03F03 701D            GOSUB  gnxtcr
4411 03F07 847     =FILEP+ ST=0   7
4412 03F0A 2F              P=     15
4413 03F0C 307             LCHEX  7             7 CHARACTER LIMIT (14 NIBS), NOT CO
```

```
4414 03F0F AC7  =FILEP! D=C    S              1ST CHARACTER
4415 03F12 AC5         B=C    S
4416 03F15 20          P=     0
4417 03F17 8E00        GOSUBL =BLANKC
           00
4418 03F1D AFA         A=C    W
4419 03F20 D5          B=C    A              Only need B field
4420            *
4421 03F22 7040        GOSUB  CATC++
4422 03F26 500         RTNNC                 NO CARRY=>NOT ALPHA NOR DIGIT
4423 03F29 861         ?ST=0  1              ALPHA?
4424 03F2C 40          GOYES  FILEP3
4425 03F2E 03          RTNCC                 CARRY CLR FOR ILLEGAL 1ST CHAR (R.E
4426            *
4427 03F30 814   FILEP3 ASRC
4428 03F33 814         ASRC
4429 03F36 171         D1=D1+ 2              POINT TO NEXT CHARACTER
4430 03F39 A4F         D=D-1  S
4431 03F3C 4C0         GOC    FILEP7
4432 03F3F 7320        GOSUB  CATC++
4433 03F43 4CE         GOC    FILEP3         FOUND ANOTHER ALPHA OR DIGIT?
4434            *
4435            *
4436            *
4437 03F46 AE4         A=B    B
4438 03F49 AC9   FILEP7 C=B    S
4439 03F4C B4B         C=C-D  S              FOR WP WRITE
4440 03F4F A46         C=C+C  S
4441 03F52 A4E         C=C-1  S
4442 03F55 400         RTNC
4443 03F58 814   FILEP9 ASRC                 SHIFT CIRCULAR FOR PADDED BLANKS
4444 03F5B 814         ASRC
4445 03F5E 964         ?A#B   B              NOT A BLANK?
4446 03F61 00          RTNYES
4447 03F63 54F         GONC   FILEP9         (B.E.T.)
4448            *
```

```
4449                    STITLE CATCHR - Categorize Character
4450                ********************************************************************
4451                ********************************************************************
4452                **
4453                ** Name:(S) CATCHR  -  Categorize Character
4454                ** Name:(S) CATCH+  -  Convert to Uppercase, Categorize Character
4455                ** Name:(S) CATC++  -  Convert to Uppercase, Categorize Character
4456                **
4457                ** Category:   PARUTL
4458                **
4459                ** Purpose:
4460                **      Categorize character in A(B) as a
4461                **      digit or letter or special character.
4462                **
4463                **      CATCH+ and CATC++ entries convert a lowercase letter to
4464                **      uppercase before categorizing it.
4465                **
4466                ** Entry:
4467                **      3 entry points:
4468                **      1) CATCHR - A(B) = character to categorize
4469                **      2) CATCH+ - A(B) = character to categorize
4470                **                  P=0
4471                **      3) CATC++ - D1 points to character to categorize.
4472                **                  P=0
4473                **
4474                ** Exit:
4475                **      P=0
4476                **      A(B)=Character that was categorized
4477                **           (a letter gets converted to uppercase
4478                **            for CATC++ and CATCH+ entries)
4479                **
4480                **      Carry set:
4481                **       Character is a digit or letter
4482                **       S1=1 iff it's a digit
4483                **
4484                **      Carry clear:
4485                **       S2=1 iff special character:  * + - . / blank
4486                **
4487                ** Calls:      CONVUC,DRANGE,ARANGE,RANGE
4488                **
4489                ** Uses:       C(A), S1,S2      - CATCHR entry
4490                **             A(B),C(A), S1,S2 - CATC++,CATCH+ entries
4491                **
4492                ** Stk lvls:   2
4493                **
4494                ** History:
4495                **
4496                **    Date     Programmer             Modification
4497                **  --------   ----------   ----------------------------------
4498                **  07/08/82   JP           Modified documentation
4499                **  09/02/82   S.W.         Changed RANGE call to DRANGE
4500                **
4501                ********************************************************************
4502                ********************************************************************
4503                *
```

```
4504 03F66 14B  =CATC++ A=DAT1 B
4505 03F69 8F00 =CATCH+ GOSBVL =CONVUC        CONVERT L.C. TO U.C.
           000
4506 03F70 841  =CATCHR ST=0   1              Clear digit found flag
4507 03F73 842          ST=0   2              Clear special char found flag
4508 03F76 20           P=     0
4509 03F78 8F00         GOSBVL =DRANGE
           000
4510 03F7F 470          GOC    CATC02         Not digit
4511 03F82 851          ST=1   1              Set Digit Found flag
4512 03F85 02    CATC01 RTNSC                 Set carry and return
4513 03F87 7000  CATC02 GOSUB  =ARANGE
4514 03F8B 59F          GONC   CATC01         Letter found
4515 03F8E 33A2         LCASC  \/*\           Load special char range: /-*
           F2
4516 03F94 7000         GOSUB  =Range
4517 03F98 5D0          GONC   CATC04         Special char, check if comma
4518 03F9B 3102         LCASC  \ \
4519 03F9F 962          ?A=C   B              Check if blank
4520 03FA2 D0           GOYES  CATC05
4521 03FA4 03           RTNCC
4522 03FA6 31C2  CATC04 LCASC  \,\
4523 03FAA 962          ?A=C   B              Comma not allowed
4524 03FAD 50           GOYES  RTNCC
4525 03FAF 852   CATC05 ST=1   2              Set Special Character flag
4526 03FB2 03   =RTNCC  RTNCC
4527            *
```

```
4528                    STITLE Uppercase Conversion
4529            ********************************************************************
4530            ********************************************************************
4531            **
4532            ** Name:    CNV2UC  -  Converts 8 chars to uppercase
4533            ** Name:(S) CNVWUC  -  Converts 8 chars to uppercase
4534            ** Name:(S) CVUCW   -  Converts 8 chars to uppercase
4535            **
4536            ** Category:   CONVRT
4537            **
4538            ** Purpose:
4539            **      Converts 8 lowercase characters to uppercase.
4540            **
4541            **      Lowercase characters are converted to uppercase by
4542            **      clearing bit 5 of the ASCII code.  All characters
4543            **      with character codes from 60-7F HEX get bit 5
4544            **      cleared.  This results in ensuring that digits,
4545            **      uppercase letters, and most special characters are
4546            **      left unchanged.  However, any character within the
4547            **      range of 60-7F that is not a lowercase letter WILL
4548            **      have its character code altered.
4549            **
4550            ** Entry:
4551            **      3 entry points:
4552            **       1) CNV2UC - D1 at possible preceding blanks before
4553            **                   characters to convert.
4554            **       2) CNVWUC - D1 at 1st character to convert.
4555            **                   P=0.
4556            **       3) CVUCW  - A contains characters to convert.
4557            **                    (it may contain any no. of characters).
4558            **                   P=0.
4559            **
4560            ** Exit:
4561            **      P=0
4562            **      Carry clear
4563            **      Every byte in A has bit 5 zeroed.
4564            **      CNV2UC:
4565            **       D1 points at the first non-blank character
4566            **       A contains the following eight bytes with
4567            **        bit 5 zeroed in every byte.
4568            **      CNVWUC:
4569            **       Same as CNV2UC, except D1 is preserved from entry.
4570            **      CVUCW:
4571            **       D1 preserved from entry
4572            **
4573            ** Calls:      GNXTCR, BLANKC
4574            **
4575            ** Uses:       A,C, D1   -  CNV2UC entry
4576            **              A,C       -  CNVWUC, CVUCW entries
4577            **
4578            ** Stk lvls:   1
4579            **
4580            ** NOTE:
4581            **      only works if characters are upper- or lower-case chars
4582            **      to begin with
```

```
4583                    **
4584                    ** History:
4585                    **
4586                    **    Date      Programmer              Modification
4587                    **  --------    ----------   ----------------------------------
4588                    **  07/08/82    S.W.         Added documentation
4589                    **
4590                    ********************************************************************
4591                    ********************************************************************
4592                    *
4593 03FB4 7F5C =CNV2UC GOSUB  gnxtcr          Get next non-blank character
4594 03FB8 1537 =CNVWUC A=DAT1 W
4595 03FBC 8E00 =CVUCW  GOSUBL =BLANKC        0010 0000 in all bytes of C
          00
4596 03FC2 A76          C=C+C  W              0100 0000 in all bytes of C
4597 03FC5 0E72         C=A&C  W              Ensure bit 5 cleared only when
4598 03FC9 81E          CSRB                   bit 6 is set
4599 03FCC 0E72         C=A&C  W              Clear all bits except bit 5
4600 03FD0 B7A          A=A-C  W              Wherever set, clr bit 5
4601 03FD3 03           RTNCC
```

```
 !CK     Ext                 -   421
 !CK2    Ext                 -  1731
 #CK     Ext                 -   112  1692  1868  2549
=ADDP    Abs   14851 #03A03  -  2132
 ARANGE  Ext                 -  1380  4513
 ARRY01  Abs   13943 #03677  -  1016   908
 ARRY02  Abs   13951 #0367F  -  1019  1015  1713
=ARRYCK  Abs   13930 #0366A  -  1012
 ASNPE+  Abs   15041 #03AC1  -  2445   476  1921  2361  2550
=ASSNP   Abs   15067 #03ADB  -  2549
=AUTOP   Abs   15363 #03C03  -  2959
 AVS=DO  Ext                 -  3818
 BLANKC  Ext                 -  4417  4595
 BLKOK   Ext                 -   673
 BLNKCK  Ext                 -   745
=CALLP   Abs   14492 #0389C  -  1840
 CALP00  Abs   14515 #038B3  -  1849  1854
 CALP01  Abs   14527 #038BF  -  1853  1848
 CALP02  Abs   14543 #038CF  -  1860  1856
 CALP03  Abs   14547 #038D3  -  1864  1844
 CALP04  Abs   14551 #038D7  -  1865  1851
 CALP05  Abs   14554 #038DA  -  1867  1894
 CALP10  Abs   14571 #038EB  -  1873  1869
 CALP15  Abs   14623 #0391F  -  1890  1928  1931
 CALP17  Abs   14627 #03923  -  1891  1919
 CALP20  Abs   14695 #03967  -  1914  1886
 CALP25  Abs   14700 #0396C  -  1916  1876
 CALP30  Abs   14705 #03971  -  1918  1915  1932
 CALP35  Abs   14717 #0397D  -  1923  1888
 CALPE1  Abs   14687 #0395F  -  1911  1882  1917
 CALPE2  Abs   14691 #03963  -  1912  1884
 CALPE3  Abs   14479 #0388F  -  1833  1837
 CALPIN  Abs   14644 #03934  -  1897  1865
 CALPNL  Abs   14483 #03893  -  1836  1841
=CATC++  Abs   16230 #03F66  -  4504  4421  4432
 CATC01  Abs   16261 #03F85  -  4512  4514
 CATC02  Abs   16263 #03F87  -  4513  4510
 CATC04  Abs   16294 #03FA6  -  4522  4517
 CATC05  Abs   16303 #03FAF  -  4525  4520
=CATCH+  Abs   16233 #03F69  -  4505  4399
=CATCHR  Abs   16240 #03F70  -  4506
=CATP    Abs   14988 #03A8C  -  2419
=CHAINP  Abs   15790 #03DAE  -  3999
 CHK-OD  Abs   14125 #0372D  -  1333   676   747
=CLKPRS  Abs   14245 #037A5  -  1629
 CLRCA+  Abs   15039 #03ABF  -  2443  2431  2439
 CLRPRM  Ext                 -  1873
=CNV2UC  Abs   16308 #03FB4  -  4593  1899
=CNVWUC  Abs   16312 #03FB8  -  4594  2358  4139
=COMCK   Abs   14029 #036CD  -  1218  2308  2880  2885
 COMCK+  Ext                 -   115   914  1445  1566  2069  2075  2311
=COMCK1  Abs   14033 #036D1  -  1219   897  1013  1893  2065  3201
 COMCKO  Ext                 -  1724  4027
=CONTP   Abs   15404 #03C2C  -  3028
 CONTP2  Abs   15410 #03C32  -  3029  2964  3093
```

```
 CONVUC  Ext                  -  4505
=COPYP   Abs   15116 #03B0C  -  2665
=CREATP  Abs   14926 #03A4E  -  2304
=CVUCW   Abs   16316 #03FBC  -  4595
=D'LTP   Abs   15048 #03AC8  -  2495
 D1C=R3  Ext                  -   942  1112  3803
 D=AVME  Ext                  -  3796
=DATACK  Abs   14046 #036DE  -  1304   749  1843
=DATP    Abs   13672 #03568  -   749   759   762
 DATP00  Abs   13659 #0355B  -   745   753   756
 DATP30  Abs   13676 #0356C  -   750   764
=DEFAP   Abs   15524 #03CA4  -  3241
=DFKEYP  Abs   14164 #03754  -  1442
=DISPP   Abs   13732 #035A4  -   873
 DRANGE  Ext                  -  1743  4509
=DROPP   Abs   14851 #03A03  -  2133
=DSPP02  Abs   13735 #035A7  -   874   868
 DSPP05  Abs   13749 #035B5  -   880   876   882   884
 DSTp    Ext                  -  2072  2077
 DSUSGP  Abs   13878 #03636  -   937   933
 DUSP20  Abs   13913 #03659  -   948   941
 DUSP30  Abs   13745 #035B1  -   878   898   901
=EDITP   Abs   15244 #03B8C  -  2854
=ELSEP   Abs   13457 #03491  -   408
 ELSEP2  Abs   13475 #034A3  -   417   448
 ELSEP5  Abs   13497 #034B9  -   446   412
 ELSEP6  Abs   13566 #034FE  -   473   450   452   454
 ELSEP7  Abs   13575 #03507  -   475   422
=ENDP    Abs   14884 #03A24  -  2240
 ENDP10  Abs   14909 #03A3D  -  2250  2242  2244
 EOLCK+  Ext                  -  2954
 EOLCKR  Ext                  -   889
 ERR01   Ext                  -   402
 ERR02   Ext                  -  1114
 ERR03   Ext                  -  1721
 ERR04   Ext                  -  2083
 ERR05   Ext                  -   424
 ERR08   Ext                  -  2966
 ERR09   Ext                  -  1317
 ERR10   Ext                  -  1023
 ERR11   Ext                  -  2441
 ERRX0   Ext                  -  4246
 EXPPAR  Ext                  -  1763
=EXTIF+  Abs   13540 #034E4  -   462   459
 EXTIFP  Abs   13525 #034D5  -   457   417
 Err03   Abs   13926 #03666  -   953   945
 FASCFD  Ext                  -  4243
=FETCHP  Abs   15389 #03C1D  -  3021
=FILEP   Abs   16028 #03E9C  -  4365  2762  3697
=FILEP!  Abs   16143 #03F0F  -  4414
=FILEP+  Abs   16135 #03F07  -  4411
=FILEP-  Abs   16128 #03F00  -  4409  2003  4375
 FILEP0  Abs   16118 #03EF6  -  4404  4398
=FILEP1  Abs   16124 #03EFC  -  4407  4391
 FILEP3  Abs   16176 #03F30  -  4427  4424  4433
```

```
 FILEP7  Abs   16201 #03F49 -  4438  4431
 FILEP9  Abs   16216 #03F58 -  4443  4447
 FILEP^  Abs   16093 #03EDD -  4394  4380  4382  4384  4386  4388
 FILPEX  Abs   16023 #03E97 -  4363  4369
 FINDA   Ext                  446
 FIXp    Ext                  194  2365
=FLIPP   Abs   15770 #03D9A -  3868
=FLTYPp  Abs   15985 #03E71 -  4241  2304  4149
=FORP    Abs   13341 #0341D -  180
 FORPE1  Abs   13450 #0348A -  402   140   190   926  1833  2446
 FORPE5  Abs   13493 #034B5 -  424   244   246   543  2084
 FORPER  Abs   13411 #03463 -  249   184
 FREE07  Ext                  249
 FREE15  Ext                  3814
=FSPC10  Abs   15593 #03CE9 -  3725  2005  2767  3735  3767
=FSPC12  Abs   15604 #03CF4 -  3728
 FSPC15  Abs   15614 #03CFE -  3733  3718
 FSPC25  Abs   15626 #03D0A -  3739  3699
 FSPC30  Abs   15634 #03D12 -  3742  3736
 FSPC35  Abs   15709 #03D5D -  3796  3758
 FSPC37  Abs   15724 #03D6C -  3803  3797
=FSPECp  Abs   15557 #03CC5 -  3697  2305  2425  2566  2671  2859  4155
 FTKYP2  Abs   15418 #03C3A -  3032  3023
 FTKYP3  Abs   15422 #03C3E -  3034  2961  3030  4050
 FTKYP5  Abs   15431 #03C47 -  3038  2956  3086
 GNXCR+  Ext                  1703  1711
 GNXTCR  Ext                  181   1304  2968
 IFCK    Ext                  2250
=IFP     Abs   13415 #03467 -  391
 IFP50   Abs   13454 #0348E -  404   399
=KEYP    Abs   14171 #0375B -  1444
 KEYP10  Abs   14208 #03780 -  1457  1452
 KYWMSG  Abs   13485 #034AD -  421   409
=LABELP  Abs   16031 #03E9F -  4366
 LBLINP  Ext                  3028  4058
 LBLNIF  Ext                  474
 LINP    Ext                  944   4040
=LISTP   Abs   15250 #03B92 -  2856
 LNP06   Ext                  419
 LPRNCK  Ext                  907
=LRP     Abs   14790 #039C6 -  2064
 LRPE4   Abs   14841 #039F9 -  2083  2066
 LRPE5   Abs   14847 #039FF -  2084  2183
 LRPe    Abs   14837 #039F5 -  2081  2074
 LSTP01  Abs   15257 #03B99 -  2859  2852
 LSTP10  Abs   15293 #03BBD -  2878  2860  2867
 LSTP12  Abs   15298 #03BC2 -  2880  3091
 LSTP20  Abs   15305 #03BC9 -  2883  2501  2874  2963  3088
 LSTP40  Abs   15319 #03BD7 -  2889  2883
 LSTPDN  Abs   15291 #03BBB -  2876  2693  2708  2879
 LSTPE   Abs   15347 #03BF3 -  2897  2892  3249
=MERGEP  Abs   15237 #03B85 -  2851
 MNPTCC  Abs   15755 #03D8B -  3816  3774  3778  3780  3799  3812
 MNPTP   Abs   15657 #03D29 -  3764  3744
 MNPTP1  Abs   15666 #03D32 -  3771  3766
```

```
 MNPTP5  Abs   15732 #03D74 -  3809  3776
=NAMEP   Abs   15214 #03B6E -  2762
 NAMEP7  Abs   15230 #03B7E -  2769  2422  2666  2700  2766  2857  2881  2886
 NTOKEN  Ext                -  1765
 NTOKNL  Ext                -  2889
=NUMC++  Abs   13968 #03690 -  1100  1012  2186  2552
=NUMC+0  Abs   13974 #03696 -  1103   114   187
=NUMCK   Abs   13981 #0369D -  1106   129   396  1016  1101  2064  2067  3200
 NUMCK1  Abs   13997 #036AD -  1112  1108  1166  1911
 NUMCK2  Abs   14001 #036B1 -  1113   924  1110  1165
 NUMCK3  Abs   14004 #036B4 -  1114  1632  1912
=NUMCKO  Abs   13977 #03699 -  1104   191  2310
 NUMS    Abs   15507 #03C93 -  3200  2136  3202
=NXTP    Abs   13397 #03455 -   243   180
 OAGNXT  Ext                -  1730
 OBFSPp  Abs   15553 #03CC1 -  3696  1907  4001
=OPSTRp  Abs   14212 #03784 -  1506
 OUT1TK  Ext                -   124
 OUT2TK  Ext                -  4248
 OUT3TK  Ext                -  2895
=OUTB+5  Abs   14390 #03836 -  1734
 OUTBYT  Ext                -  2016
=OUTLI1  Abs   14089 #03709 -  1319  1313
 OUTLI2  Abs   14095 #0370F -  1321  1328
=OUTLIT  Abs   14067 #036F3 -  1312  1307
=OUTNB4  Abs   14394 #0383A -  1735
 OUTNBS  Ext                -  3729
 OUTNIB  Ext                -  1717
=OUTVAR  Abs   14142 #0373E -  1379   544  1384  1702  1723  1860
=OUTbyt  Abs   14782 #039BE -  2015
=PILP    Abs   13270 #033D6 -   111   867
=PILP+   Abs   13273 #033D9 -   112
 PILP10  Abs   13305 #033F9 -   124   132  1331
 PILP30  Abs   13309 #033FD -   126   116
 PILP40  Abs   13327 #0340F -   134   951
 PILP80  Abs   13333 #03415 -   137   127
 PILP85  Abs   13338 #0341A -   140   123
=POKEP   Abs   14222 #0378E -  1565
 POKEP9  Abs   14241 #037A1 -  1571  1446  1455
 POLLD+  Ext                -  3819
=PRENCK  Abs   13953 #03681 -  1020  1726  1895  2187
 PRGP30  Abs   15028 #03AB4 -  2438  2435
 PRGP80  Abs   15033 #03AB9 -  2441  1909  2306  2568  2676  2709  4052
 PRGPE1  Abs   15044 #03AC4 -  2446  2555  2702  4144
=PRIVTP  Abs   15234 #03B82 -  2849
 PRMCNT  Ext                -  1878
 PRT#P   Abs   13839 #0360F -   917   870
 PRT#P0  Abs   13832 #03608 -   914   918   920
=PRTP    Abs   13721 #03599 -   867
 PRTP20  Abs   13792 #035E0 -   897   910
 PRTP70  Abs   13807 #035EF -   903   890   915   935
 PRTP80  Abs   13811 #035F3 -   905   886
 PRTP85  Abs   13851 #0361B -   922   895
 PRTPE1  Abs   13860 #03624 -   926   906  1568
=PURGEP  Abs   14985 #03A89 -  2418
```

PURGP+	Abs	14999	#03A97	-	2425	2420					
R.STPR	Ext			-	460						
=R3=D1*	Abs	13602	#03522	-	600						
=R3=D1+	Abs	13618	#03532	-	606						
=R3=D10	Abs	13606	#03526	-	601	1762					
=R3=D1C	Abs	13612	#0352C	-	604						
=RANDP	Abs	14958	#03A6E	-	2358						
RANDP2	Abs	14905	#03A39	-	2248	2070	2076	2079	2134	2309	2312	2364
REDPE5	Ext			-	2081						
=REMP	Abs	13640	#03548	-	673						
=REMP05	Abs	13649	#03551	-	675						
=REMP10	Abs	13652	#03554	-	676	674	677				
=RENAMP	Abs	15132	#03B1C	-	2671	2668					
=RENMP	Abs	15435	#03C4B	-	3085						
RENMP1	Abs	15463	#03C67	-	3094	3090					
=RESETp	Abs	15465	#03C69	-	3139						
RESPTR	Ext			-	135	600	892	903	1571		
=RETRNp	Abs	15458	#03C62	-	3092						
RNMP05	Abs	15129	#03B19	-	2670	2682	2696				
RNMP15	Abs	15158	#03B36	-	2684	2678					
RNMP20	Abs	15172	#03B44	-	2690	2685					
RNMP25	Abs	15176	#03B48	-	2692	2672	2688				
RNMP45	Abs	15201	#03B61	-	2706	2436	2690				
RNMP65	Abs	15210	#03B6A	-	2709	2763	2863	2870	2873		
=RTNCC	Abs	16306	#03FB2	-	4526	4524					
=RUNP	Abs	15793	#03DB1	-	4000						
RUNP10	Abs	15823	#03DCF	-	4025	4003					
RUNP20	Abs	15870	#03DFE	-	4058	4028					
RUNP30	Abs	15858	#03DF2	-	4049	4026	4059				
RUNP40	Abs	15862	#03DF6	-	4050	4037	4041				
RUNPE3	Abs	15866	#03DFA	-	4052	4015	4017	4167	4178	4181	
RUNPE8	Abs	15377	#03C11	-	2966	3036					
Range	Ext			-	3716	4516					
ResPtr	Abs	13329	#03411	-	135	119	196				
Resptr	Abs	15520	#03CA0	-	3203	3143	3710	3875	4038	4407	
SBNME1	Abs	14713	#03979	-	1921	2004					
SBNMP0	Abs	14759	#039A7	-	2006	1864					
SBNMP3	Abs	14778	#039BA	-	2014	2009					
=SECURP	Abs	15247	#03B8F	-	2855						
=SFLGP	Abs	15483	#03C7B	-	3193						
=STATP	Abs	14862	#03A0E	-	2182						
STLXPT	Ext			-	4242						
STMTL0	Ext			-	467						
=STRGCK	Abs	14010	#036BA	-	1159	1444	1448	1565	1569	3032	
STRGer	Abs	14021	#036C5	-	1164	1161					
=STRNGP	Abs	14237	#0379D	-	1569	1507	1567				
SUBNMP	Abs	14748	#0399C	-	2003	1689	1850				
=SUBP	Abs	14258	#037B2	-	1688						
SUBP10	Abs	14269	#037BD	-	1692	1725					
SUBP15	Abs	14322	#037F2	-	1712	1707					
SUBP20	Abs	14352	#03810	-	1723	1699					
SUBP25	Abs	14356	#03814	-	1724	1718	1747	1753	1759		
SUBP30	Abs	14367	#0381F	-	1728	1690					
SUBP35	Abs	14394	#0383A	-	1736	1688	1732	1758			
SUBP45	Abs	14400	#03840	-	1740	1693					

```
 SUBP55  Abs   14425 #03859 -  1749  1744
 SUBP60  Abs   14441 #03869 -  1755  1751
 SUBPE3  Abs   14346 #0380A -  1720  1697  1757
 SVRST+  Ext                -   410
 TFMP10  Abs   15903 #03E1F -  4148  4180  4194
 TFMP15  Abs   15921 #03E31 -  4154  4142
 TFMP20  Abs   15924 #03E34 -  4155  4151
 TFMP60  Abs   15960 #03E58 -  4190  4165
 TFMP70  Abs   15983 #03E6F -  4198  4191
=TRSFMP  Abs   15883 #03E0B -  4139
 UPDIN+  Ext                -   468
=USERP   Abs   15770 #03D9A -  3869
=USINGp  Abs   13864 #03628 -   931   875
=VARP    Abs   13582 #0350E -   541   243  2182
=VARP05  Abs   13586 #03512 -   542
 WRDSC+  Ext                -   192   397
 WRDSCN  Ext                -  1767
 WSR0-3  Ext                -   394
 ^ERR01  Abs   15899 #03E1B -  4144  4196
 ^ERR1   Abs   15896 #03E18 -  4143  4060
 a"      Ext                -  1305
 a'      Ext                -   760  1308
 d1c=r3  Abs   15724 #03D6C -  3802  3755  4371
 eFTYPE  Ext                -  4245
 eolck+  Abs   15351 #03BF7 -  2954   408  1506  1840  2133  2363  2419  2665
                               2856  2960  3085  4150
 err03   Abs   14346 #0380A -  1721   953  1709  2897
=exppar  Abs   14463 #0387F -  1763   893   917  1629
 fixp    Abs   14981 #03A85 -  2365  2313
 gnxtcr  Abs   15383 #03C17 -  2968  1761  3809  4241  4410  4593
 ntoken  Abs   14467 #03883 -  1765   541   909   947  1218  1694  1741  1846
                               2006  3029  4049
 out1t+  Abs   14118 #03726 -  1330   763  1336  1740  1871  3813
 out1tk  Abs   14121 #03729 -  1331   878  1104  1321  1325  1338  1379  1443
                               1746  2556
 out2tk  Abs   16017 #03E91 -  4248  1752  4244
 outbyt  Abs   14782 #039BE -  2016  1457  1734  1891  1898  3696  3726
 outnbs  Abs   15608 #03CF8 -  3729  1737  2255
 pDEVCp  Abs       1 #00001 -    11  3795
 pFSPCp  Abs       4 #00004 -    11  3757
=polld+  Abs   15757 #03D8D -  3818  3756  3794
 r3exp+  Abs   14455 #03877 -  1761  1106  1159
 r3expp  Abs   14459 #0387B -  1762   939  1874  4366
 resptr  Abs   14241 #037A1 -  1570  1508  1631  1838  1849  2011  2248  2500
                               2565  2769  3026  3038  3199  3203  3793  4152
 rtncc   Abs   14788 #039C4 -  2062  1903  2185  2246  2497  2562  3141  3195
                               3197  3243  3245  3247  3871  3873
 sRFILE  Abs       8 #00008 -    11  4002  4018  4036
 t*      Ext                -  2561
 tALL    Ext                -  2245  2429  2496  3194  4381
 tCARD   Ext                -  2433  2686  3777  4385
 tCLOCK  Abs 1376751 #501EF -    11  3140
 tCOMMA  Ext                -   185   881  1220  2893
 tCREF   Ext                -  1918
 tCVAL   Ext                -  1890
```

tDEF	Ext			-	2241						
tEOL	Ext			-	1733						
tEXTIF	Ext			-	464						
tEXTND	Abs	2490863	#601EF	-	11	3246					
tFN	Ext			-	447						
tIN	Ext			-	1905						
tINT2	Ext			-	1749						
tINTO	Abs	3015151	#E01EF	-	11	4176	4193	4387			
tKEY	Ext			-	3022						
tKEYS	Ext			-	2706	2865	4383				
tLINE#	Ext			-	449	2890					
tLITRL	Ext			-	3725						
tMAIN	Ext			-	3773						
tMATH	Abs	3539439	#601EF	-	11	3196					
tOFF	Ext			-	3244	3872					
tON	Ext			-	3242	3870					
tPCRD	Abs	4063727	#E01EF	-	11	3779					
tPORT	Ext			-	3775						
tPRMEN	Ext			-	1728	1897					
tPRMST	Ext			-	2014						
tRFILE	Ext			-	4000						
tSEMIC	Ext			-	121	130	883	948	1453		
tSTEP	Ext			-	193						
tSUB	Ext			-	2243						
tTAB	Ext			-	885						
tTHEN	Ext			-	398						
tTO	Ext			-	188	2553	2680	2695	4379		
tUSING	Ext			-	932						
wrdscn	Abs	14473	#03889	-	1767	880	931	2240	2495	2560	2694	3021
					3139	3193	3241	3772	3869	4192	4377	

Input Parameters

  Source file name is JP&PR3::MS

  Listing file name is JP/PR3:TI:ML::-1

  Object file name is JP%PR3:TI:MS::-1

                                    111111
                          0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
1            *     SSS   BBBB    &     EEEEE  X   X  PPPP
2            *    S   S  B   B  & &    E      X   X  P   P
3            *    S      B   B  & &    E       X X   P   P
4            *     SSS   BBBB    &     EEEE     X    PPPP
5            *        S  B   B  & & &  E       X X   P
6            *    S   S  B   B  &  &   E      X   X  P
7            *     SSS   BBBB    && &  EEEEE  X   X  P
8            *
9                    TITLE  Expression Parser<831212.1206>
10 03FD5             ABS    #03FD5
11           ****************************************************************
12           ****************************************************************
13           **
14           ** Name:(S) EXPPAR  -  Expression Parse
15           ** Name:(S) EXPPLS  -  Expression Parse for Left of Equal Sign
16           ** Name:(S) EXPP10  -  Expr Parse (specify start of parse stk)
17           **
18           ** Category:   PARUTL
19           **
20           ** Purpose:
21           **      Parse an expression and compile correct code for it
22           **      Also parses dummy array references
23           **      EXPPLS will stop parsing when a valid left-hand-side
24           **      has been found.
25           **
26           ** Entry:
27           **      D0 is pointer to output stream
28           **      D1 is pointer to input stream
29           **      EXPPLS requires LeftSd(S7) to be set on entry.
30           **      EXPP10 requires LeftSd(S7) to be clear on entry and
31           **         that D(A) be set to where the parse stack should
32           **         start.
33           **
34           ** Exit:
35           **      If dummy array found then
36           **      Carry set
37           **      S0 -- 1 (invalid expression)
38           **      S1 -- Set by last NTOKEN
39           **      S2 -- Set by last NTOKEN
40           **      S3 -- 1 (not valid string expression)
41           **      S7 -- Clear if EXPPAR, unchanged if EXPPLS
42           **      D0 -- Points past code compiled for dummy array
43           **      D1 -- Points past first token not used in expression
44           **      A  -- Contains first token not used for dummy array
45           **      P  -- 0
46           **      XM -- 0
47           **
48           **      else
49           **      Carry clear
50           **      S0 -- 0 if valid expression found, 1 otherwise
51           **      S1 -- Set by last NTOKEN
52           **      S2 -- Set by last NTOKEN
53           **      S3 -- 0 if valid string expr. found, 1 otherwise
54           **      S7 -- Clear if EXPPAR, unchanged if EXPPLS
55           **      D0 -- Points past code compiled for expression
```

```
56          **      D1 -- Points past first token not used in expression
57          **      A  -- Contains first token not used in expression
58          **      XM -- Set iff expression is clearly a value expr
59          **      P  -- 0
60          **      D(A) -- (MTHSTK)
61          **      (PRMCNT) set non-zero if expression contained user FN
62          **
63          ** Calls:      NTOKEN, OUT1TK, OUTNIB, OUTVAR, OUTLIT, OUTBYT,
64          **             RANGE, CMPB#C, SCAN, DELET1, DELET2, LOOK, LOOK2
65          **             GNXCR+, OUTNBS, PARMCK, BOPCOM, CONCOM, PUSH-P,
66          **             PUSH-3, INSRT1, RESPTR, CKLFSD
67          **
68          ** Uses:     A,B,C,D(15-5),R0,R1,S0,S1,S2,S3,S7,S11,Carry
69          **           FUNCD0,PRMCNT(first nib)
70          **
71          **
72          ** Stk lvls:   3
73          **
74          ** Detail:
75          **      Internal representation of non-terminals is:
76          **         00 -- Primary
77          **         01 -- S-expr
78          **         02 -- Factor
79          **         03 -- Term
80          **         04 -- Sum
81          **         05 -- Relation
82          **         06 -- Conjunction
83          **         07 -- Expression
84          **         08 -- N-func-ref
85          **         09 -- S-func-ref
86          **         0A -- Substring ref
87          **         0C -- StartR (Reference expression)
88          **         0D -- StartS (Reference expression w/substring)
89          **         0E -- StartV (Value expression)
90          **
91          **      This parser is essentially a stack automaton.
92          **      The stack builds from high memory down to lower
93          **      memory.  All stack elements are 2 bytes (4 nibs)
94          **      in length although 2 or more elements may be used
95          **      to hold extra information if needed.
96          **
97          **      If EXPPLS is called with LeftSd set, the parser will
98          **      stop when it sees an reference expression or a
99          **      substring reference expression followed by an equals
100         **      sign.
101         **
102         **      Code is compiled from low memory toward high memory.
103         **      The code pointer and the stack pointer are checked
104         **      to make sure they never collide.  MEMERR is called
105         **      if there is such a collision.
106         **
107         **      Value expressions are indicated upon return by the
108         **      XM bit.  This is used to determine whether a parameter
109         **      in a CALL statement is a reference or value parameter.
110         **      It is also used to determine whether an expression
```

```
111             **      would be a valid destination address for an assignment
112             **      such as the INPUT statement.
113             **
114             ************************************************************************
115             ************************************************************************
```

```
                    EJECT
                *  The following is a loosely interpreted explanation
                *  of the stack automata that is implemented.
                *  To interpret it, states are identified by a symbol
                *  followed by a colon.  Items on the stack are enclosed in
                *  angle brackets (< >).  On the right side, a plus sign
                *  indicates that some code needs to be compiled.  An asterisk
                *  on the right side indicates that the next token in the
                *  stream should be place on top of the stack.  If the stack
                *  is to be modified, the next line is used to show the
                *  way the stack should be changed to look;  this is indicated
                *  by an arrow (-->).  If no arrow is shown, then the stack is
                *  not modified before continuing.
                *  A symbol is shown on the far right
                *  that is the next state for the parser--think of this as a
                *  GO TO.
                *
                *  Basically, the parser starts in state E0, which is the
                *  operand (expression) expected state.  The stack is compared
                *  with the patterns shown (in the order shown) for a match.
                *  If the pattern matchs, the parser modifies the stack as
                *  suggested by the table, compiles code (if necessary) and
                *  scans for another token and places it on stack (if
                *  necessary) and goes to the next state.
                *
                *  The expression parser terminates with an error if it gets
                *  to the end of the patterns for a given state without
                *  matching.  An error consists of getting to the ACCEPT
                *  state with the stack containing something other than
                *  a single numeric or string expression on top of the stack.
                *
                * E0:  <unary op>  . . . . . .  * E0
                *      (           . . . . . .  * E0
                *      <number>    . . . . . . + * P1
                *                  --> <primary>
                *      <var>       . . . . . .  * AR0
                *      <func name> . . . . . .  * FR0
                *      <string var>. . . . . .  * SE0
                *      <string literal>  . . . + * SE1
                *                  --> <s-expr>
                *      <s-func-name>  . . . . .  * SF0
                *
                * P1:  <factor> ^ <primary> <??>  . . . . + F1
                *                  --> <factor> <??>
                *      <primary> <??>  . . . . .    F1
                *                  --> <factor> <??>
                *
                * F1:  <unary op> <factor> <??>  . . . . + P1
                *                  --> <primary> <??>
                *      <factor> ^  . . . . . .  * E0
                *      <term> <mulop> <factor> <??>  . . + T1
                *                  --> <term> <??>
                *      <factor> <??>  . . . . .    T1
                *                  --> <term> <??>
                *
```

```
171          * T1:  <term> <mulop>  . . . . . .  * E0
172          *      <sum> <term>  <??>  . . . . . +  SUM1
173          *                  --> <sum> <??>  . . . . .
174          *      <term> <??>  . . . . . .  SUM1
175          *                  --> <sum> <??>
176          *
177          * SUM1:<sum> <unary op (not NOT)>  . . . . .  * E0
178          * SUM2:<relation> <relop> <sum> <??>  . . . . +  REL1
179          *                  --> <relation> <??>
180          *      <sum> <??>  . . . . . .  REL1
181          *                  --> <relation> <??>
182          *
183          * REL1:<relation> <relop>  . . . . . .  * E0
184          *      <conjunction> AND <relation> <??>  . . . +  CON1
185          *                  --> <conjunction> <??>
186          *      <relation> <??>  . . . . . .  CON1
187          *                  --> <conjunction> <??>
188          *
189          * CON1:<conjunction> AND  . . . . . .  * E0
190          *      <expr> OR <conjunction> <??>  . . . . +  E1
191          *                  --> <expr> <??>
192          *      <conjunction> <??>  . . . . . .  E1
193          *                  --> <expr> <??>  . . . . .
194          *
195          * E1:  <expr> OR  . . . . . .  * E0
196          *      ( <expr> )  . . . . . .  * P1
197          *                  --> <primary>
198          *      ( <expr> ,  . . . . . .  * E0
199          *                  --> <Cmplx> <func-ref>
200          * E2:  <func-name> <func-ref> <(s|n)-expr> )  . . +  * P1
201          *                  --> <primary>
202          *      <func-ref> <(s|n)-expr> ,  . . . . . +  * E0
203          *                  --> <func ref>
204          *      <func-name> <s-func-ref> <(s|n)-expr> )  .  * SE1
205          *                  --> <s-expr>
206          *      <s-func-ref> <(s|n)-expr> ,  . . . . . +  * E0
207          *                  --> <s-func-ref>
208          *      <substr-ref> <(s|n)-expr> ,  . . . . . +  * E0
209          *                  --> <substr-ref>
210          *      <substr-ref> <(s|n)-expr> ]  . . . . . +  * SE1
211          *                  --> <s-expr>
212          *      <(s|n)-expr> <??>
213          *                  --> (ACCEPT!!)
214          *
215          * AR0: <var/ASC> (  . . . . . .  * E0
216          *                  --> <func name/ASC> <func ref>
217          *      <var> <??>  . . . . . .  P1
218          *                  --> <primary> <??>
219          *
220          * FR0: <func name> (  . . . . . .  * E0
221          *                  --> <func name><func ref>
222          *      <func name> <??>  . . . . . .  P1
223          *                  --> <primary> <??>
224          *
225          * SE0  <string var> (  . . . . . .  * E0
```

```
                *                    --> <array> <s-func-ref>
                *      <string var> <??>  . . . . . + SE1
                *                    --> <s-expr> <??>  . . . .
                *
                * SE1: <s-expr> [  . . . . . * E0
                *                    --> <substr-ref>  . . . .
                *      <s-expr> & <s-expr> <??>  . . . . . + SE2
                *                    --> <s-expr> <??>  . . . .
                *
                * SE2: <s-expr> &  . . . . . * E0
                *      <s-expr> <relop> <s-expr> <??>  . . . . + SUM2
                *                    --> <relation> <??>  . . .
                *      <s-expr> <relop>  . . . . . * E0
                *      ( <s-expr> )  . . . . . * SE1
                *                    --> <s-expr>
                *      <s-expr> <??>  . . . . . E2
                *
                * SF0: <s-func-name> (  . . . . . * E0
                *                 --> <func name/ASC> <s-func-ref>
                *      <s-func-name> <??>  . . . . . + * SE1
                *                    --> <s-expr> <??>
                
                 NPrim  EQU   #00          Parse stack token definition
                 SExpr  EQU   #01          Parse stack token definition
                 Factor EQU   #02          Parse stack token definition
                 Term   EQU   #03          Parse stack token definition
                 Sum    EQU   #04          Parse stack token definition
                 Relat  EQU   #05          Parse stack token definition
                 Conjun EQU   #06          Parse stack token definition
                 NExpr  EQU   #07          Parse stack token definition
                 NFuncR EQU   #08          Parse stack token definition
                 SFuncR EQU   #09          Parse stack token definition
                 SubstR EQU   #0A          Parse stack token definition
                 StartR EQU   #0C          Parse stack token definition
                 StartS EQU   #0D          Parse stack token definition
                 StartV EQU   #0E          Parse stack token definition
                        RDSYMB SB%TAB::MS
                =FRange EQU   (LASTFN)~#6A Range of functions
                 InvalE EQU   0            Status bit definition
                 String EQU   1            Status bit definition
                 ArgLst EQU   2            Status bit definition
                 NumExp EQU   3            Status bit definition
                 LeftSd EQU   7            Status bit definition
                 VarFlg EQU   11           Status bit definition
                 t=     EQU   2~tRELOP
                *-
03FD5 6000       OUTBYj GOTO  =OUTbyt
                *-
                *-
03FD9 847        =EXPPAR ST=0  LeftSd      Don't look for left-hand-side
03FDC 8F00       =EXPPLS GOSBVL =D=AVME
      000
03FE3 840        =EXPP10 ST=0  InvalE      Valid expression
                                           No characterization nibble
03FE6 821               XM=0               (Reference expression)
```

```
280 03FE9 20          P=     0
281 03FEB 31C0        LC(2)  StartR          Start (Reference Expression)
282 03FEF AFA         A=C    W               Copy start token to A
283 03FF2 7416  E0-10 GOSUB  SCAN            Scan for first item
284             *
285             *   Looking for the start of an expression
286 03FF6 109   E0    R1=C                   Save lex restart info
287 03FF9 3782        LCHEX  81828728        t-,tNOT,t+,(
          7828
          18
288 04003 27          P=     7
289 04005 8F00        GOSBVL =MEMBER         Is it one of these things?
          000
290 0400C 55E         GONC   E0-10           Yes, then SCAN; GOTO E0
291 0400F 7EB5        GOSUB  ARANGE          Is it in range [A,Z]?
292 04013 5F0         GONC   E0+05           Yes, then compile variable
293 04016 3306        LC(4)  (tADIG9)~(tADIG0)   Alpha Digit range
          96
294 0401C 77B5        GOSUB  Range           Is it in range?
295 04020 4C0         GOC    E0+10           No, then continue
296 04023 8E00  E0+05 GOSUBL =OUTVAR         Compile variable
          00
297 04029 6791  P1-10j GOTO  P1-10
298             *-
299             *-
300 0402D       E0+10
301 0402D 7226        GOSUB  CHKCON          Check for constant
302 04031 490         GOC    E0+20           Jump if no constant found
303 04034 79E7        GOSUB  RTNSXM          Set Value Flag
304 04038 50F         GONC   P1-10j          (B.E.T.) Scan; goto P1
305             *-
306             *-
307 0403B 31D2  E0+20 LC(2)  tSVAR           String vars
308 0403F 966         ?C#A   B               Is it a string variable?
309 04042 C0          GOYES  E0+30           No, then continue
310 04044 8E00        GOSUBL =OUTVAR         Compile string variable
          00
311 0404A 6D14        GOTO   SE1-10
312             *-
313             *-
314 0404E 8E00  E0+30 GOSUBL =QUOTCK         Is it a quote?
          00
315 04054 531         GONC   E0+40           No, then continue
316 04057 8E00        GOSUBL =OUTLI1         Compile literal
          00
317 0405D 4D6         GOC    E0+41           Error?  (No closing quote)
318 04060 7DB7        GOSUB  RTNSXM          Set Value Flag
319 04064 6304        GOTO   SE1-10
320             *-
321             *-
322 04068 33C7  E0+40 LC(4)  (tARRAY)~(tFN)
          D7
323 0406E 7565        GOSUB  Range           Is it in range?
324 04072 5C5         GONC   E0+41.          Yes, then handle function ref
325 04075 313B        LC(2)  tXFN            External Function
```

```
    326 04079 962          ?A=C   B          Is it an XFN?
    327 0407C C5           GOYES  E0+42      Yes, then handle it
    328 0407E E6           C=C+1  A          Assumes FFN=XFN+1
    329             *      LC(2)  tFFN       Funny function
    330 04080 966          ?A#C   B          Is it a funny function
    331 04083 E0           GOYES  E0+40.     No, then skip
    332 04085 27           P=     7
    333 04087 7DE5         GOSUB  outnbs     Compile 8 nibbles (FFN DD DD Len)
    334 0408B D9           C=B    A          Prepare to jump to parse handler
    335 0408D 06           RSTK=C
    336 0408F 03           RTNCC             Jump to parse handler
    337             *-
    338             *-
    339 04091 3308  E0+40. LCHEX  8F80       Range of binary/unary operators
              F8
    340 04097 7C35         GOSUB  Range      Is it in range?
    341 0409B 5F2          GONC   E0+41      Yes, then error
    342 0409E 33A6         LC(4)  FRange     Range of Functions
              4B
    343 040A4 23           P=     3          One item will be needed on stack
    344 040A6 7D25         GOSUB  Range      Is it in range?
    345 040AA 5F2          GONC   E0+44      Yes, then handle function ref
    346 040AD 20           P=     0
    347 040AF 31FE         LC(2)  tXWORD     XWORD token
    348 040B3 966          ?A#C   B
    349 040B6 51           GOYES  E0+41
    350 040B8 8E00         GOSUBL =RESPTR    Restore D1
              00
    351 040BE 111          A=R1              Recall lex info
    352 040C1 8E00         GOSUBL =PRESCN    Resume lexing
              00
    353 040C7 6E2F  E0j    GOTO   E0
    354             *-
    355             *-
    356 040CB 6344  E0+41  GOTO   ACCEPT     No, then error (expr expected)
    357             *-
    358             *-
    359 040CF 96A   E0+41. ?C=0   B          Is it an FN?
    360 040D2 60           GOYES  E0+42      No, then skip
    361 040D4 7157         GOSUB  SETPRM
    362 040D8 27    E0+42  P=     7          Two stack items
    363 040DA 7205  E0+44  GOSUB  PUSH-P     Push items
    364 040DE CD           B=B-1  A
    365 040E0 CD           B=B-1  A          Point at arg range (EXAD-2)
    366 040E2 D4           A=B    A
    367 040E4 BF1          BSL    W
    368 040E7 BF1          BSL    W
    369 040EA BF1          BSL    W          B(7-3)=ExecAd-2
    370 040ED 862          ?ST=0  ArgLst     Is there an argument list?
    371 040F0 C4           GOYES  E0+52      No, then compile now
    372 040F2 3180         LC(2)  NFuncR
    373 040F6 861          ?ST=0  String     Is it a string function?
    374 040F9 50           GOYES  E0+46      No, then leave it numeric
    375 040FB 309          LC(1)  SFuncR     Yes, then change to string func
    376 040FE AF4   E0+46  A=B    W
```

```
377 04101 AEA          A=C    B            Copy function ref token
378 04104 27           P=     7
379 04106 76D4         GOSUB  PUSH-P       Push 2 items
380 0410A 8E00         GOSUBL =GNXCR+      Skip paren and get next char
          00,
381 04110 D2           C=0    A
382 04112 7D84         GOSUB  LOOK
383 04116 3192         LCASC  \)\
384 0411A 87B          ?ST=1  Varflg       Is it an array ref?
385 0411D 73           GOYES  E0+60        Yes, then check for dummy array
386 0411F 966          ?A#C   B
387 04122 61           GOYES  E0+48
388 04124 7A54         GOSUB  DELET1
389 04128 171          D1=D1+ 2            Skip past Rparen
390 0412B AF9          C=B    W
391 0412E 8F00         GOSBVL =CSRW3       Shift C right 3 times
          000
392 04135 5F0          GONC   E0+54        (B.E.T.)
393             *-
394             *-
395 04138 6827  E0+48  GOTO   DUMARY       Check for dummy array as a parm
396             *-
397             *-
398 0413C 85B   E0+52  ST=1   Varflg       Don't compile tLPRP when done
399 0413F 7B94         GOSUB  PUSH-3       Push dummy item on stack
400 04143 DE           ACEX   A            Move exec addr-2 to C
401 04145 136   E0+54  CDOEX
402 04148 14A          A=DAT0 B            Read arg range
403 0414B 134          D0=C                Restore D0
404 0414E F0           ASL    A            A(2)=max, A(1)=min
405 04150 68E1         GOTO   E1+20        Fall into Func compile code
406             *-
407             *-
408 04154 D1    E0+60  B=0    A            Clear arg count
409 04156 962          ?A=C   B            Is it a Rparen?
410 04159 22           GOYES  E0+64        Yes, then okay
411 0415B 31C2         LCASC  \,\          No, then
412 0415F 966          ?A#C   B            Is it a comma?
413 04162 6D           GOYES  E0+48        No, then look for an expression
414 04164 871          ?ST=1  String
415 04167 65           GOYES  E0+68        String arrays cannot have 2 args
416 04169 8E00         GOSUBL =GNXCR+      Get next char
          00
417 0416F 3192         LCASC  \)\
418 04173 966          ?A#C   B            Is it a Rparen?
419 04176 74           GOYES  E0+68        No, then error, must have (,)
420 04178 B05          B=B+1  P            Increment arg count
421 0417B B05   E0+64  B=B+1  P            Increment arg count
422 0417E 171          D1=D1+ 2            Skip past Rparen
423 04181 D4           A=B    A
424 04183 308          LCHEX  8            Look(3)
425 04186 7914         GOSUB  LOOK
426 0418A AF9          C=B    W
427 0418D 28           P=     8
428 0418F 31C0         LC(2)  StartR       Check for start token
```

```
429 04193 29          P=     9
430 04195 915         ?B#C   WP          Is dummy array first in expr?
431 04198 52          GOYES  E0+68       No, then error
432 0419A AFC         ABEX   W
433 0419D 20          P=     0
434 0419F B04         A=A+1  P           Change Array token to Dummy Array
435 041A2 8E00        GOSUBL =OUTVAR
            00
436 041A8 AF9         C=B    W           Copy arg count
437 041AB 8E00        GOSUBL =OUTNIB     Compile arg count
            00
438 041B1 7000        GOSUB  =NTOKEN
439 041B5 850         ST=1   InvalE      Set invalid expression
440 041B8 853         ST=1   NumExp      Never invalid string expression
441 041BB 02          RTNSC              Return from expression parse
442             *-
443             *-
444 041BD 6153  E0+68 GOTO   ACCEPT
445             *-
446             *-
```

```
447                     EJECT
448                 ****************************************************************
449                 ****************************************************************
450                 **
451                 ** Name:(S) P1-10   -  Numeric Operand Found
452                 **
453                 ** Category:   PARUTL
454                 **
455                 ** Purpose:
456                 **      Point of reentry for numeric funny functions
457                 **
458                 ** Entry:
459                 **      P      =  0
460                 **      D0   = Output ptr (points past last nib of FFN code)
461                 **      D(A) = Stack pointer
462                 **      D1   = Input ptr (points past last char in FFN text,
463                 **                 which is probably the closing paren)
464                 **      If a funny function is re-entering here, it should
465                 **      have set the XM bit to indicate that a value expression
466                 **      has been parsed.
467                 **
468                 ** NOTE:
469                 **      At this point a numeric operand has just been compiled.
470                 **      Funny functions are a special type of function that
471                 **      allow the expression parser to be extended to include
472                 **      that have special parse and/or execution requirements.
473                 **      See IDS for a complete description of how to implement
474                 **      a funny function.
475                 **
476                 ** History:
477                 **
478                 **    Date     Programmer             Modification
479                 **  --------   ----------   ------------------------------------
480                 **  09/27/83   B.S.         Added documentation
481                 **
482                 ****************************************************************
483                 ****************************************************************
484 041C1 AE0   =P1-10  A=0    B               Primary
485 041C4 7244          GOSUB  SCAN
486 041C8 3408  P1      LC(5)  (Factor)~(t^)~(t^)
            082
487 041CF 7C16          GOSUB  BOPCOM          Check for:
488                                            <Factor>  <^> <Primary> <??>
489 041D3 5A0           GONC   F1              Found, then skip
490 041D6 3120          LC(2)  Factor          Factor
491 041DA 7EA3          GOSUB  INSRT1          Insert factor
492 041DE       F1
493 041DE 3108          LC(2)  t^              Uparrow
494 041E2 966           ?A#C   B               Is it an uparrow?
495 041E5 60            GOYES  F1+05           No, then continue
496 041E7 6A0E          GOTO   E0-10           Yes, then SCAN; goto E0
497                 *-
498                 *-
499 041EB 71B3  F1+05   GOSUB  LOOK2           LOOK(2)
500 041EF 3318          LC(4)  (t-)~(tNOT)     Minus, NOT?
```

```
          28
501 041F5 AEC          ABEX   B
502 041F8 7BD3         GOSUB  Range          Is it in range?
503 041FC 422          GOC    F1+20
504 041FF 7914         GOSUB  out1tk
505 04203 AE4          A=B    B
506 04206       F1+10
507               *   Disallow  <^> <Unary op> <Factor>
508 04206 3108         LC(2)  t^
509 0420A 70B3         GOSUB  CMPB#C
510 0420E 5EA          GONC   E0+68
511 04211 7806         GOSUB  DL1SXM
512 04215 AE2          C=0    B
513 04218 7073         GOSUB  INSRT1
514 0421C 5BA          GONC   P1             (B.E.T.)
515               *-
516               *-
517 0421F 3178  F1+20  LC(2)  t+
518 04223 AEC          ABEX   B
519 04226 961          ?B=C   B
520 04229 DD           GOYES  F1+10
521 0422B 3438         LC(5)  (Term)~(tDIV)~(t*)
          683
522 04232 79B5         GOSUB  BOPCOM         Check for:
523                                          <Term> <Mulop> <Factor> <??>
524               *    GONC   T1             (Optional)
525 04236 3130         LC(2)  Term           Term
526 0423A 7E43         GOSUB  INSRT1
527 0423E 3338  T1     LC(4)  (tDIV)~(t*)      Mulop range
          68
528 04244 7F83         GOSUB  Range          Is it in range?
529 04248 5C6          GONC   E0-J0          Yes, then SCAN; goto E0
530 0424B 7153         GOSUB  LOOK2
531 0424F 3140         LC(2)  Sum
532 04253 965          ?B#C   B
533 04256 41           GOYES  SUM-10
534 04258 3178         LC(2)  t+             Plus
535 0425C 757D         GOSUB  OUTBYj         Compile add
536 04260 AEA          A=C    B              Restore parsetoken
537 04263 76B5         GOSUB  DL1SXM
538 04267 560          GONC   SUM1           (B.E.T.)
539               *-
540               *-
541 0426A 7E13  SUM-10 GOSUB  INSRT1
542 0426E 3178  SUM1   LC(2)  t+             Plus
543 04272 962          ?A=C   B              Is it a plus?
544 04275 04           GOYES  E0-J0          Yes, then Scan; goto E0
545 04277 302          LC(1)  t-             Minus
546 0427A 962          ?A=C   B              Is it a minus?
547 0427D 83           GOYES  E0-J0          Yes, then Scan; goto E0
548 0427F 34A8         LC(5)  (Relat)~(tRELOP)~(tRELOP)
          A85
549 04286 7565         GOSUB  BOPCOM         Check for:
550                                          <Rel> <Relop> <Sum> <??>
551 0428A 421          GOC    REL-05         If not found,
```

```
552                                              change <Sum> to <Rel>
553 0428D D9    SUM+10 C=B    A                 If found then compile third
554 0428F F6           CSR    A                     nibble of relop
555 04291 F6           CSR    A
556 04293 D8           B=A    A
557 04295 8E00         GOSUBL =OUTNBC           Compile third nibble of relop
          00
558 0429B D4           A=B    A
559 0429D 3150  REL-05 LC(2)  Relat
560 042A1 77E2         GOSUB  INSRT1
561 042A5 31A8  REL1   LC(2)  tRELOP            Relop
562 042A9 966          ?A#C   B                 Is it a relop?
563 042AC D0           GOYES  REL+10            No, then continue
564 042AE 7A85         GOSUB  CKLFSD            Check for left-hand-side
565 042B2 560          GONC   REL+10            Found, then don't accept =
566 042B5 6C3D  E0-J0  GOTO   E0-10             Yes, then scan; goto E0
567             *-
568             *-
569 042B9 34B8  REL+10 LC(5)  (Conjun)~(tAND)~(tAND)
          B86
570 042C0 7B25         GOSUB  BOPCOM            Check for:
571                                              <Conj> <AND> <Rel> <??>
572 042C4 5A0          GONC   CON1
573 042C7 3160         LC(2)  Conjun            Conjunction
574 042CB 7DB2         GOSUB  INSRT1
575 042CF 31B8  CON1   LC(2)  tAND              And
576 042D3 962          ?A=C   B                 Is it an and?
577 042D6 FD           GOYES  E0-J0             Yes, then scan; goto E0
578 042D8 34C8         LC(5)  (NExpr)~(tOR)~(tEXOR)
          D87
579 042DF 7C05         GOSUB  BOPCOM            Look for:
580                                              <NExpr> <OR> <Conj> <??>
581             *      GONC   E1                (Optional)
582 042E3 3170  E1-10  LC(2)  NExpr             Expr
583 042E7 71A2         GOSUB  INSRT1
584 042EB 853   E1     ST=1   NumExp
585 042EE 33C8         LC(4)  (tOR)~(tEXOR)     Disjunctor range
          D8
586 042F4 7FD2         GOSUB  Range             Is it an OR or EXOR?
587 042F8 5CB          GONC   E0-J0             Yes, then scan; goto E0
588 042FB 3192  E1+03  LCASC  \)\               Right paren
589 042FF 962          ?A=C   B                 LOOK(0)=Right paren?
590 04302 60           GOYES  E1+05             Yes, then skip
591 04304 66C0         GOTO   E1+30             No, then continue
592             *-
593             *-
594 04308 7492  E1+05  GOSUB  LOOK2             LOOK(2)
595 0430C 3182         LCASC  \(\               Left paren
596 04310 965          ?B#C   B                 LOOK(2)=Left paren?
597 04313 71           GOYES  E1+10             No,continue
598 04315 7005         GOSUB  DL2SXM
599 04319 863          ?ST=0  NumExp            Is it a numeric expression?
600 0431C 60           GOYES  E1+08
601 0431E 62AE  P1-J2  GOTO   P1-10             Insert(Primary); Scan; goto P1
602             *-
```

```
603                *-
604 04322 6541  E1+08  GOTO   SE1-10          Insert(SExpr); Scan; goto SE1
605                *-
606                *-
607 04326 68E1  E1+09  GOTO   ACCEPT
608                *-
609                *-
610 0432A 78A4  E1+10  GOSUB  FCNTYP          Set status bit for string
611                                           or numeric fcn
612 0432E 7A44  E1+15  GOSUB  PARMCK          Increment parm count, check type
613 04332 7842         GOSUB  DELET2          Delete Func Ref and Expr
614 04336 85B          ST=1   VarFlg          No LPRP following function
615 04339       E1+20
616 04339 BB0          ASL    X
617 0433C 9A0          ?B<A   XS             Too few parms?
618 0433F 7E           GOYES  E1+09           Yes, then error
619 04341 AF4   FCNCOM A=B    W              Save actual arg count in A(XS)
620                *  Stack looks like:       <Intrinsic> <Exad>  <??>
621                *                 or    <Info>  <Array>  <Exad>  <??>
622                *                 or    <Info>  <FN>     <Exad>  <??>
623                *                 or    <Info>  <XFN>    <Exad>  <??>
624                *                                          ^
625                *                                          |
626                *                       Garbage if no parameters |
627 04344 7852         GOSUB  LOOK2           Get function descriptor
628 04348 71D4         GOSUB  DL1SXM          Delete Exad or other garbage
629 0434C AFC          ABEX   W               Return actual arg count to B(XS)
630 0434F 32D7         LC(3)  #200+tARRAY     Array token, C(XS)=2
          2
631 04354 966          ?A#C   B               Is it an array reference?
632 04357 A2           GOYES  FCNC20          No, then continue
633 04359 861          ?ST=0  String          Is it a string array?
634 0435C 70           GOYES  FCNC05          No, then okay
635 0435E 921          ?B=C   XS              Yes, then does it have 2 dims
636 04361 5C           GOYES  E1+09           Yes, then error
637                                           (>2 dims already trapped)
638 04363 AF6   FCNC05 C=A    W               Copy word to C
639 04366 28           P=     8
640 04368 31C0         LC(2)  StartR          Change 2 nibbles
641 0436C 29           P=     9               Check if
642 0436E 916          ?A#C   WP                 <StartR> just above <Array>
643 04371 50           GOYES  FCNC10          No, then okay
644 04373 821          XM=0                   Yes, then reset Value Flag
645 04376 20    FCNC10 P=     0
646 04378 8E00  FCNC15 GOSUBL =OUTVAR         Compile FN or ARRAY ref
          00
647 0437E 5B3          GONC   FCNC42          (B.E.T.) Now compile arg count
648                *-
649                *-
650 04381 30C   FCNC20 LC(1)  tFN             FN token
651 04384 962          ?A=C   B               Is it a user FN reference?
652 04387 1F           GOYES  FCNC15          Yes, then compile FN ref
653 04389 313B         LC(2)  tXFN            XFN token
654 0438D 962          ?A=C   B               Is it an XFN token?
655 04390 42           GOYES  FCNC40          Yes, then compile XFN ref
```

```
656 04392 7682          GOSUB  out1tk          No, then compile intrinsic func
657 04396 78E1          GOSUB  DELET1          Delete Intrinsic descriptor
658 0439A 87B   FCNC30  ?ST=1  VarFlg         Is a () required following fcn
659 0439D A0            GOYES  FCNC32          No, then okay
660 0439F 31AA          LC(2)  =tLPRP          Yes,
661 043A3 7E2C          GOSUB  OUTBYj          then compile LPRP
662 043A7 871   FCNC32  ?ST=1  String         Is it a string function?
663 043AA 60            GOYES  FCNC35          Yes, then goto SE1-10
664 043AC 641E          GOTO   P1-10           No, then goto P1-10
665             *-
666             *-
667 043B0 67B0  FCNC35  GOTO   SE1-10          Goto SE1-10
668             *-
669             *-
670 043B4 25    FCNC40  P=     5               Compile 6 nibbles
671 043B6 7EB2          GOSUB  outnbs          Compile XFN
672 043BA F5    FCNC42  BSR    A
673 043BC F5            BSR    A
674 043BE D4            A=B    A
675 043C0 74B2          GOSUB  outnbs          Compile argument count
676 043C4 76B1  FCNC45  GOSUB  DELET2          Delete Function Descriptor
677 043C8 51D           GONC   FCNC30          (B.E.T.)
678             *-
679             *-
680 043CB        E2
681 043CB 71D1  E1+30   GOSUB  LOOK2
682 043CF 311F          LC(2)  tCOMMA          Comma
683 043D3 966           ?C#A   B               Is it a comma
684 043D6 55            GOYES  E1+35           No, then continue
685 043D8 AFC           ABEX   W
686 043DB 3380          LC(4)  (SubstR)~(NFuncR)  Range[N,S-func, substr]
          A0
687 043E1 72F1          GOSUB  Range           Is it a function ref, etc?
688 043E5 AFC           ABEX   W
689 043E8 5C2           GONC   E1+32           Yes, then okay
690 043EB 3182          LCASC  \(\             No,...
691 043EF 965           ?B#C   B               ... then do we have  "(Expr,"?
692 043F2 53            GOYES  Accept          No, then error
693 043F4 7681          GOSUB  DELET2          Delete  (  and  Expr
694
695 043F8 3D            NIBHEX 3D              LC(14)
696 043FA 0000           CON(4)   0                Dummy stack entry
697 043FE 80             CON(2)   NFuncR           < NFuncR>
698 04400 0              CON(1)   0                Parm count
699 04401 0000           CON(5)   (=CMPLX)-2       Exec ad - 2
          0
700 04406 A7             CON(2)   tCMPLX           CMPLX token
701
702 04408 2F            P=     15
703 0440A AFA           A=C    W
704 0440D 7FC1          GOSUB  PUSH-P          Push 16 nibbles (4 items)
705                                              onto stack
706 04411 7B81          GOSUB  LOOK2           Prepare for PARMCK
707 04415 7363  E1+32   GOSUB  PARMCK          Increment parm count & ck type
708 04419 7561          GOSUB  DELET1          Delete <E>
```

```
    709 0441D D9          C=B    A            Recall Func ref w/new arg count
    710 0441F 7961        GOSUB  INSRT1       Update parameter count
    711 04423 6344        GOTO   DUMAO5       Check if next arg is dummy array
    712           *-
    713           *-
    714
    715 04427 67E0  Accept GOTO  ACCEPT
    716           *-
    717           *-
    718 0442B 31D5  E1+35  LCASC  \]\
    719 0442F 966         ?A#C   B            Is it a ]?
    720 04432 5F          GOYES  Accept       No, then try to accept expr
    721 04434 31A0        LC(2)  SubstR       Yes, then check for Substr Ref
    722 04438 965         ?B#C   B            Is it a substring?
    723 0443B CE          GOYES  Accept       No, then try to accept
    724 0443D 7B33        GOSUB  PARMCK       Increment arg count
    725 04441 AA9         C=B    XS           Copy arg count
    726 04444 317A        LC(2)  tISUB$       Implied substring token
    727 04448 22          P=     2            Compile 3 nibbles
    728 0444A 8E00        GOSUBL =OUTNBC      Compile substring ref
              00
    729 04450 7A21        GOSUB  DELET2       Delete <E> <SubstR>
    730 04454 7841        GOSUB  LOOK2
    731 04458 7621        GOSUB  DELET1       Delete <Exad>
    732 0445C 31D0        LC(2)  StartS
    733 04460 965         ?B#C   B            Is this alone on stack?
    734 04463 50          GOYES  SE1-10       No, then skip
    735 04465 821         XM=0                Yes, then clear Value Flag
```

```
                        EJECT
                *****************************************************************
                *****************************************************************
                **
                ** Name:(S) SE1-10  -  String Operand Found
                **
                ** Category:   PARUTL
                **
                ** Purpose:
                **      Point of reentry for string funny functions
                **
                ** Entry:
                **      P      = 0
                **      D0   = Output ptr (points past last nib of FFN code)
                **      D(A) = Stack pointer
                **      D1   = Input ptr (points past last char in FFN text,
                **                 which is probably the closing paren)
                **      If a funny function is re-entering here, it should
                **      have set the XM bit to indicate that a value expression
                **      has been parsed.
                **
                ** NOTE:
                **      At this point a numeric operand has just been compiled.
                **      Funny functions are a special type of function that
                **      allow the expression parser to be extended to include
                **      that have special parse and/or execution requirements.
                **      See IDS for a complete description of how to implement
                **      a funny function.
                **
                ** History:
                **
                **    Date     Programmer            Modification
                **  --------   ----------   ----------------------------------
                **  09/27/83   B.S.         Added documentation
                **
                *****************************************************************
                *****************************************************************
04468 AE0   =SE1-10 A=0    B
0446B B04           A=A+1  P               String expression
0446E 7891          GOSUB  SCAN
04472 31B5  SE1     LCASC  \[\
04476 966           ?A#C   B               Is it a [?
04479 95            GOYES  SE1+20          No, then continue
0447B DB            C=D    A
0447D 136           CDOEX
04480 163           D0=D0+ 4               Point to Look(2)
04483 14A           A=DAT0 B               Read byte
04486 134           D0=C                   Restore D0
04489 31D0          LC(2)  StartS
0448D 962           ?A=C   B               Is it a StartS item?
04490 21            GOYES  SE1+05          Yes, then change it to StartV
04492 30C           LC(1)  StartR
04495 966           ?A#C   B               Is it a StartR item?
04498 B1            GOYES  SE1+10          No, then okay
0449A 831           ?XM=0                  Yes, then is Value Flag now set?
```

```
791 0449D 50          GOYES  SE1+05        No, then change to StartS
792 0449F B04         A=A+1  P             Yes, then change to StartV
793 044A2 B04  SE1+05 A=A+1  P             Change Start token
794 044A5 DB          C=D    A
795 044A7 136         CDOEX
796 044AA 163         DO=DO+ 4             Get start token address
797 044AD 148         DAT0=A B             Write out changed start token
798 044B0 134         DO=C                 Restore DO
799 044B3 3400  SE1+10 LC(5)  (=SUB$)-2     Load address of SUB$ exec routine
          000
800 044BA F6          CSR    A             Throw away lowest digit
801 044BC 7CC0        GOSUB  INSRT1        Put these upper 4 digits on stack
802 044C0 23          P=     3             Load lowest digit now
803 044C2 300         LC(1)  (=SUB$)-2
804 044C5 20          P=     0
805 044C7 32A0        LC(3)  SubstR        Fill in SubstR token
          0
806 044CC DA          A=C    A             Move to A to store on stack
807 044CE 632B  E0-J2 GOTO   E0-10         Scan; goto E0
808             *-
809             *-
810 044D2 3498  SE1+20 LC(5)  (SExpr)~(t&)~(t&)
          981
811 044D9 7213        GOSUB  BOPCOM        Check for:
812                                        <SExpr> <&> <SExpr> <??>
813 044DD 3198        LC(2)  t&            Check for: <SExpr> <&>
814 044E1 962         ?A=C   B             Found?
815 044E4 AE          GOYES  E0-J2         Yes, Scan; goto E0
816 044E6 34A8        LC(5)  (SExpr)~(tRELOP)~(tRELOP)
          A81
817 044ED 7EF2        GOSUB  BOPCOM        Check for:
818                                        <SExpr> <relop> <SExpr> <??>
819 044F1 591         GONC   SUM+j         If found then fall into REL code
820 044F4 843         ST=0   NumExp
821 044F7 31A8        LC(2)  tRELOP        Relop
822 044FB 966         ?A#C   B             Is it a relop?
823 044FE 90          GOYES  E1+03j        No, fall into num exp context
824 04500 7833        GOSUB  CKLFSD        Yes, check for left-hand-side
825 04504 49C         GOC    E0-J2         Found? No, then stack operator
826 04507 63FD  E1+03j GOTO   E1+03        Fall into num exp context code
827             *-
828             *-
829 0450B 618D  SUM+j  GOTO   SUM+10
830             *-
831             *-
832             ******************************************************************
833             ******************************************************************
834             **
835             ** Name:(S) ACCEPT  -  Funny function parse error reentry point
836             **
837             ** Category:   PARUTL
838             **
839             ** Purpose:
840             **      This is the point where funny function parse routines
841             **      should reenter if they detect an error
```

```
842                **
843                ** Entry:
844                **      D(A) is stack pointer
845                **      A(W) set by last call to NTOKEN (flexible, doesn't
846                **          matter if an error is being flagged)
847                **      D0 is output pointer (flexible, doesn't matter if
848                **          an error is being flagged)
849                **      D1 is input pointer should point past first token
850                **          not used in expressin (flexible, doesn't matter
851                **          if an error is being flagged).
852                **      Status bits set by last NTOKEN call (or equivalent)
853                **
854                ** Exit:
855                **      See exit conditions for EXPRDC
856                **
857                ** History:
858                **
859                **    Date      Programmer              Modification
860                **  --------   ----------   ----------------------------------
861                **  11/01/83   B.S.         Added documentation
862                **
863                ******************************************************************
864                ******************************************************************
865 0450F 853    =ACCEPT ST=1    NumExp          Numeric expression
866 04512 20           P=      0               Need P to be cleared
867 04514 D6           C=A     A               Save parse token
868 04516 06           RSTK=C
869 04518 133          AD1EX
870 0451B 1F00         D1=(5)  =AVMEME
            000
871 04522 147          C=DAT1  A               Read in end of stack pointer
872 04525 131          D1=A
873 04528 DF           CDEX    A               Move curr stk ptr to C(A)
874 0452A 136          CDOEX
875 0452D 14A          A=DAT0  B               LOOK(1)
876 04530 163          DO=DO+  4               Delete one stack item
877 04533 AE8          B=A     B
878 04536 14A          A=DAT0  B
879 04539 134          DO=C                    Restore pointer
880 0453C 3170         LC(2)   NExpr           Expression
881 04540 961          ?B=C    B               Is there an n-expr on stack
882 04543 D0           GOYES   ACPT10          Yes, valid numeric expr found
883 04545 301          LC(1)   SExpr           String expression
884 04548 965          ?B#C    B               Is there an s-expr on stack?
885 0454B D1           GOYES   ACPT20          No, error in expression
886 0454D 843          ST=0    NumExp          Valid string expr found
887 04550 30E   ACPT10 LC(1)   StartV          Start of stack marker
888 04553 962          ?A=C    B               Is it a Value expression?
889 04556 E1           GOYES   ACPT30          Yes, then okay
890 04558 30C          LC(1)   StartR          Start of stack marker
891 0455B 962          ?A=C    B               Is it a Reference expression
892 0455E A1           GOYES   ACPT40          Yes, then okay
893 04560 30D          LC(1)   StartS          Start of stack marker
894 04563 962          ?A=C    B               Is is a Ref exp with substring
895 04566 21           GOYES   ACPT40
```

```
    896 04568 850   ACPT20 ST=1   InvalE          Invalid expression
    897 0456B 853          ST=1   NumExp         Never invalid string expr
    898 0456E 8E00         GOSUBL =RESPTR         Restore pointer
              00
    899 04574 79A2  ACPT30 GOSUB  RTNSXM
    900 04578 07    ACPT40 C=RSTK
    901 0457A DA           A=C    A
    902 0457C 03           RTNCC                  Return from expression parse
```

```
903                     EJECT
904                 ****************************************************************
905                 ****************************************************************
906                 **
907                 ** Name:    DELET1  -  Delete one item from parse stack
908                 ** Name:    DELET2  -  Delete two items from parse stack
909                 **
910                 ** Category:   LOCAL
911                 **
912                 ** Purpose:
913                 **      Modify parste stack pointer so as to delete one (or
914                 **      two) items.
915                 **
916                 ** Entry:
917                 **      D(A) contains parse stack pointer
918                 **
919                 ** Exit:
920                 **      D(A) contains modified parse stack pointer
921                 **
922                 ** Calls:      DELET2 calls DELET1
923                 **
924                 ** Uses.......
925                 **  Inclusive: D(A)
926                 **
927                 ** Stk lvls:   DELET1: 0,  DELET2: 1
928                 **
929                 ** Detail:
930                 **      Adds 4 to contents of D(A), no check is made to
931                 **      determine if stack has underflowed.
932                 **
933                 ** History:
934                 **
935                 **    Date     Programmer              Modification
936                 **  --------   ----------   ----------------------------------
937                 **  10/28/83   B.S.         Updated documentation
938                 **
939                 ****************************************************************
940                 ****************************************************************
941 0457E 7000   DELET2 GOSUB  DELET1         Delete 1 token (then another)
942 04582 E7     DELET1 D=D+1  A
943 04584 E7            D=D+1  A
944 04586 E7            D=D+1  A
945 04588 E7            D=D+1  A             Add 4 to pointer (pop 1 item)
946 0458A 03            RTNCC                Return with carry clear
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:    INSRT1  -  Replace item on top of stack
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Modify the first from top of parse stack by writing
                **      the contents of C(3-0) into it.
                **
                ** Entry:
                **      D(A) contains pointer to first from top of parse
                **      stack.
                **
                ** Exit:
                **      Stack item is modified.  Carry is clear.
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: None
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **   --------   ----------   ----------------------------------
                **   10/04/83   B.S.         Updated documentation
                **
                ********************************************************************
                ********************************************************************
0458C DE        INSRT1 ACEX   A              Move data item to A
0458E DF               CDEX   A              Move stack pointer to C
04590 136              CDOEX  A              Move stack pointer to DO
04593 1583             DATO=A 4              Rewrite stack item
04597 136              CDOEX                 Restore DO
0459A DF               CDEX   A              Move stack pointer to D(A)
0459C DE               ACEX   A              Restore A
0459E 03               RTNCC
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:     LOOK    -  Look at parse stack
                ** Name:     LOOK2   -  Look at parse stack
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Retrieve the Nth parse stack item below the top
                **      where N is greater than zero.  Actually the N+1th
                **      item below the top is also returned in B(7-4) and
                **      the Nth item in B(3-0)
                **
                ** Entry:
                **      P      =  0
                **      D(A) contains pointer to parse stack item first from
                **      top.
                **
                **      LOOK assumes C(0) equals the desired item
                **      number multiplied by 4 (i.e. 0 ==> LOOK(1),
                **      4 ==> LOOK(2), etc).  LOOK2 loads a 4 in orger to
                **      do a look(2).
                **
                ** Exit:
                **      B contains the desired stack items
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: B(W),C(W)
                **
                ** Stk lvls:   0
                **
                ** Detail:
                **      Reads two parse stack items into the lower 8 nibbles
                **      of the B register.  Since the items are stored
                **      backwards in memory and each item is reversed within
                **      itself, the result comes out forward.
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ----------------------------------
                **  10/04/83   B.S.         Updated documentation
                **
                ****************************************************************
                ****************************************************************
1038 045A0 304   LOOK2  LCHEX  4              Constant 4 implies LOOK(2)
1039 045A3 D1    LOOK   B=0    A              Clear address field
1040 045A5 A85         B=C    P              Copy in constant to be added
1041 045A8 DB          C=D    A              Copy stack pointer to C
1042 045AA C9          C=C+B  A              Add offset to stack pointer
1043 045AC 136         CDOEX                 Move stack address to DO
```

```
1044 045AF AFC          ABEX   W            Prepare to read
1045 045B2 1527         A=DATO W            Read four stack items
1046 045B6 AFC          ABEX   W            Restore A, stack items to B
1047 045B9 136          CDOEX               Restore DO
1048 045BC 03           RTNCC
```

```
1049                    EJECT
1050                ********************************************************************
1051                ********************************************************************
1052                **
1053                ** Name:    CMPB#C  -  Shift B and Compare to C(B)
1054                **
1055                ** Category:   GENUTL
1056                **
1057                ** Purpose:
1058                **      Shifts B right 4 nibbles then compare B(B) with C(B)
1059                **      and sets the carry if they are not equal
1060                **
1061                ** Entry:
1062                **      C(B) contains value to be compared to B(B) after
1063                **      B(W) has been shifted right 4 times.
1064                **
1065                ** Exit:
1066                **      B(W) has been shifted 4 times.
1067                **      Carry set if B(B)#C(B) on return
1068                **
1069                ** Calls:      None
1070                **
1071                ** Uses.......
1072                **  Inclusive: B(W)
1073                **
1074                ** Stk lvls:   None
1075                **
1076                ********************************************************************
1077                ********************************************************************
1078 045BE BF5    =CMPB#C BSR    W
1079 045C1 BF5            BSR    W
1080 045C4 BF5            BSR    W
1081 045C7 BF5            BSR    W               Shift over one token
1082 045CA 965            ?C#B   B               Perform comparison
1083 045CD 00             RTNYES                 Return w/carry set if not equal
1084 045CF 01             RTN                    Return w/carry clear if equal
```

```
1085                    EJECT
1086                ********************************************************************
1087                ********************************************************************
1088                **
1089                ** Name:    ARANGE  -  Checks for ASCII A thru Z
1090                **
1091                ** Category:   GENUTL
1092                **
1093                ** Purpose:
1094                **      Check if A(B) in range of ASCII A thru Z
1095                **
1096                ** Entry:
1097                **      P=0
1098                **      A(B) is byte to be checked
1099                **
1100                ** Exit:
1101                **      Carry set if not in specified range
1102                **
1103                ** Calls:      None
1104                **
1105                ** Uses.......
1106                **  Inclusive: C(A)
1107                **
1108                ** Stk lvls:   None
1109                **
1110                ********************************************************************
1111                ********************************************************************
1112 045D1 3314 =ARANGE LCASC  \ZA\
              A5
1113 045D7 8D00 =Range  GOVLNG =RANGE
              000
```

```
1114                    EJECT
1115            ****************************************************************
1116            ****************************************************************
1117            **
1118            ** Name:    SCAN    -  Push item on parse stack and call NTOKEN
1119            ** Name:    PUSH-1  -  Push item on parse stack
1120            ** Name:    PUSH-P  -  Push items on parse stack
1121            **
1122            ** Category:   LOCAL
1123            **
1124            ** Purpose:
1125            **      SCAN:
1126            **        Push the top of parse stack [A(3-0)] into RAM and
1127            **        call NTOKEN to get a new parse item onto the top of
1128            **        the stack.
1129            **      PUSH-1:
1130            **        Push one item on parse stack
1131            **      PUSH-P:
1132            **        Push word-through-pointer on stack
1133            **
1134            ** Entry:
1135            **      D(A) contains pointer to top stack item in memory.
1136            **      SCAN:
1137            **        A(3-0) contains the top of stack.
1138            **      PUSH-1:
1139            **        A(3-0) contains data to be pushed.
1140            **      PUSH-P:
1141            **        A(P-0) contains data to be pushed.
1142            **
1143            ** Exit:
1144            **      A new stack item is added to stack.
1145            **      D(A) updated to new top of stack
1146            **
1147            **      SCAN:
1148            **         Checks for a collision with the code being compiled
1149            **           before the new item is added.
1150            **         See NTOKEN for other exit conditions
1151            **
1152            ** Calls:      PUSH-3,NTOKEN
1153            **
1154            ** Uses.......
1155            **  Inclusive: Same as NTOKEN,D(A)
1156            **
1157            ** Stk lvls:   Same as NTOKEN
1158            **
1159            ** Detail:
1160            **      Decrements D(A) four times to make room for new item
1161            **      and checks for a collision.  If not, it stores the
1162            **      old top of stack item into RAM and goes to NTOKEN
1163            **
1164            ** History:
1165            **
1166            **    Date     Programmer            Modification
1167            **  --------   ----------   ----------------------------------
1168            **  10/25/83   B.S.         Updated documentation
```

```
1169            **
1170            ************************************************************************
1171            ************************************************************************
1172 045DE 23   =PUSH-3 P=     3
1173 045E0 D2   =PUSH-P C=0    A
1174 045E2 809          C+P+1
1175 045E5 E3           D=D-C  A
1176 045E7 136          CDOEX
1177 045EA 8BF          ?C>=D  A
1178 045ED 71           GOYES  COLLIS
1179 045EF DF           CDEX   A
1180 045F1 136          CDOEX
1181 045F4 1501         DATO=A WP
1182 045F8 136          CDOEX
1183 045FB DF           CDEX   A
1184 045FD 136          CDOEX
1185 04600 20           P=     0
1186 04602 03           RTNCC
1187 04604 8C00  COLLIS GOLONG =MEMERR
          00
1188 0460A 70DF  SCAN   GOSUB  PUSH-3
1189 0460E 6000         GOTO   =NTOKEN
1190            *-
1191            *-
```

```
1192                    EJECT
1193                ****************************************************************
1194                ****************************************************************
1195                **
1196                ** Name:(S) CONCOM  -  Compile a Numeric Constant
1197                **
1198                ** Category:   PARUTL
1199                **
1200                ** Purpose:
1201                **      Compiles a numeric constant (Single digit, Long Int or
1202                **      Long Real)
1203                **
1204                ** Entry:
1205                **      D0 is output pointer
1206                **      A,B set by NTOKEN
1207                **      D(A) = (AVMEME)
1208                **      P = 0
1209                **
1210                ** Exit:
1211                **      Carry clear if constant found, set otherwise
1212                **      P = 0
1213                **
1214                ** Calls:      DRANGE,OUT1TK,OUTNBS,RANGE
1215                **
1216                ** Uses.......
1217                **  Inclusive: A(W),B(W),C(W)
1218                **
1219                ** Stk lvls:   1
1220                **
1221                ****************************************************************
1222                ****************************************************************
1223                *-
1224 04612 8F00  CONC20 GOSBVL =DRANGE
           000
1225 04619 400         RTNC
1226 0461C 8C00  out1tk GOLONG =OUT1TK
           00
1227                *-
1228                *-
1229 04622 136   SMALL  CDOEX
1230 04625 1B00        D0=(5) =UNFNIB
           000
1231 0462C 1564        C=DAT0 S
1232 04630 134         D0=C
1233 04633 BF4         ASR    W
1234 04636 BE4         ASR    B
1235 04639 21          P=     1
1236 0463B B04         A=A+1  P
1237 0463E 20          P=     0
1238 04640 A4E         C=C-1  S
1239 04643 A4E         C=C-1  S
1240 04646 415         GOC    UNDF:1          Underflow trap=1
1241 04649 A4E         C=C-1  S
1242 0464C 465         GOC    UNDF:2          Underflow trap=2
1243 0464F 6141        GOTO   ParmEr
```

```
1244                *-
1245                *-
1246 04653 3111  CHKCON LC(2)  =tSMALL
1247 04657 962          ?A=C   B
1248 0465A 8C           GOYES  SMALL
1249 0465C CE           C=C-1  A
1250 0465E 962          ?A=C   B
1251 04661 B5           GOYES  BIG
1252 04663 3321         LC(4)  (tFLT1)~(tFLT12)
           D1
1253 04669 7A6F         GOSUB  Range
1254 0466D 401          GOC    CONCOM
1255 04670 7710  FLTCOM GOSUB  CONC10
1256 04674 22           P=     2
1257 04676 D4           A=B    A
1258 04678 8C00  outnbs GOLONG =OUTNBS
           00
1259                *-
1260                *-
1261 0467E 3320 =CONCOM LC(4)  (tINT2)~(tINT12)
           C0
1262 04684 7F4F         GOSUB  Range
1263 04688 498          GOC    CONC20
1264 0468B D6    CONC10 C=A    A
1265 0468D B8E          C=-C-1 P
1266 04690 80D0         P=C    0
1267 04694 63EF         GOTO   outnbs
1268                *-
1269                *-
1270 04698 3103  UNDF:1 LCASC  \0\
1271 0469C 7539         GOSUB  OUTBYj
1272 046A0 501          GONC   UNDF10          (B.E.T.)
1273                *-
1274                *-
1275 046A3       UNDF:2
1276 046A3 32D1         LCHEX  01D             One digit float=0?
           0
1277 046A8 932          ?A=C   X               Has this num denormalized to 0?
1278 046AB DE           GOYES  UNDF:1          Yes, then change to zero
1279 046AD 7FBF         GOSUB  FLTCOM
1280 046B1 3300  UNDF10 LC(4)  =eUNFLW
           00
1281 046B7 20           P=     =UNP            Parameter for =HNDL20 (Underflow)
1282                                           P=1
1283
1284 046B9 514          GONC   CONWRN          (B.E.T.)
1285                *-
1286                *-
1287 046BC 136   BIG    CDOEX
1288 046BF 1B00         D0=(5) =OVFNIB
           000
1289 046C6 1564         C=DAT0 S
1290 046CA 134          D0=C
1291 046CD A4E          C=C-1  S
1292 046D0 A4E          C=C-1  S
```

```
1293 046D3 471          GOC    OVFL:1
1294 046D6 R4E          C=C-1 S
1295 046D9 4A0          GOC    OVFL:2
1296 046DC 64B0         GOTO   ParmEr
1297            *-
1298            *-
1299 046E0 632F  MemErr GOTO   COLLIS
1300            *-
1301            *-
1302 046E4 3107  OVFL:2 LC(2)  =tINF
1303 046E8 460          GOC    CHKC40          (B.E.T.)
1304 046EB 31C6  OVFL:1 LC(2)  =tMAXRL
1305 046EF 72E8  CHKC40 GOSUB  OUTBYj
1306 046F3 3300         LC(4)  =eOVFLW
          00
1307 046F9 20           P=     =OVP             Parameter for =HNDL20  (Overflow)
1308                                            P=2
1309
1310 046FB       CONWRN
1311 046FB 108          R0=C                    Save warning number
1312
1313 046FE DB           C=D    A
1314 04700 D5           B=C    A                B(A)= parse stk ptr
1315 04702 D2           C=0    A
1316 04704 8F00         GOSBVL =HNDL20          Set flags (uses C(W),A(A),D(X))
          000
1317
1318 0470B 07           C=RSTK                  Need this subroutine level back
1319                                            will jump to P1-10 when done
1320 0470D 137          CD1EX                   C(A)=D1
1321 04710 109          R1=C                    Save D1
1322 04713 169          D0=D0+ 10               Move over space to hold
1323                                               AVMEMS& AVMEME
1324 04716 136          CD0EX
1325 04719 8B5          ?C>B   A                Is there enough memory?
1326 0471C 4C           GOYES  MemErr           No, then error out
1327 0471E 1B00         D0=(5) =AVMEMS
          000
1328 04725 15A9         A=DAT0 10               Read AVMEMS&AVMEME
1329 04729 144          DAT0=C A                AVMEMS points past this save area
1330 0472C 135          D1=C
1331 0472F 1C9          D1=D1- 10               Point at save area
1332 04732 1599         DAT1=A 10               Save AVMEMS&AVMEME
1333 04736 164          D0=D0+ 5                Point at AVMEME
1334 04739 D9           C=B    A                C=Parse stack pointer
1335 0473B 144          DAT0=C A                This becomes AVMEME
1336 0473E 8E00         GOSUBL =R<RST2          Save 3 stack levels
          00
1337 04744 118          C=R0                    Recall warning number
1338 04747 8E00         GOSUBL =MFWRQ8          Display warning
          00
1339 0474D 8E00         GOSUBL =RST2<R          Restore 3 stack levels
          00
1340 04753 8F00         GOSBVL =D=AVME          Restore parse stack pointer
          000
```

```
1341 0475A 1F00          D1=(5) =AVMEMS
           000
1342 04761 143           A=DAT1 A
1343 04764 130           D0=A
1344 04767 189           D0=D0- 10           Point at save area
1345 0476A 15A9          A=DAT0 10           Read AVMEMS&AVMEME
1346 0476E 1599          DAT1=A 10           Restore AVMEMS&AVMEME
1347 04772 119           C=R1                Recall D1
1348 04775 135           D1=C                Restore D1
1349 04778 684A          GOTO   P1-10        Now rejoin parse
```

```
1350                    EJECT
1351                ****************************************************************
1352                ****************************************************************
1353                **
1354                ** Name:    PARMCK  -  Check number and type of parameters
1355                **
1356                ** Category:   LOCAL
1357                **
1358                ** Purpose:
1359                **      Increment parameter count and check parameter type
1360                **
1361                ** Entry:
1362                **      B(7-0)=XXXXXCFF
1363                **          where XXXXX is exec addr-2,
1364                **                C     is argument count and
1365                **                FF    is either SFuncR,NFuncR or SubstR
1366                **      ST(NumExp) = 0 iff parameter is numeric
1367                **
1368                ** Exit:
1369                **      B(XS) is incremented to reflect new arg count
1370                **      Doesn't return if arg count overflows or parameter
1371                **      type doesn't match--pops stack and jumps to ACCEPT
1372                **      instead.
1373                **      R0=B(W)
1374                **      A(2)=max parms, A(1)=min parms
1375                **
1376                ** Calls:      GETDSC
1377                **
1378                ** Uses.......
1379                **  Inclusive: A(X),C,R1
1380                **
1381                ** Stk lvls:   1
1382                **
1383                ** Detail:
1384                **      The parameter count is incremented and checked
1385                **      for being either greater than 15 or greater than
1386                **      the maximum number of parameters allowed for that
1387                **      function.  The parameter descriptor nibble for that
1388                **      parameter number for that function is used to
1389                **      determine if that type (string or numeric) is
1390                **      permitted.  These descriptor nibbles are coded such
1391                **      that the highest bit is set if a numeric parameter
1392                **      is allowed and the next highest bit is set if a
1393                **      string parameter is allowed.  At least one of these
1394                **      bits must be set.
1395                **
1396                ** History:
1397                **
1398                **    Date     Programmer              Modification
1399                **  --------   ----------   ----------------------------------
1400                **  10/04/83   B.S.         Updated documentation
1401                **
1402                ****************************************************************
1403                ****************************************************************
1404 0477C 7710  PARMCK GOSUB  GETDSC         Get descriptor nibble
```

```
1405 04780 401          GOC    ParmEr
1406 04783 873          ?ST=1  NumExp          Is current parameter numeric?
1407 04786 50           GOYES  PARM10          Yes, then okay
1408 04788 A46          C=C+C  S               No, shift over descriptor bits
1409 0478B A46   PARM10 C=C+C  S               Test high bit
1410 0478E 400          RTNC                   Return if parameter type okay
1411 04791 07    ParmEr C=RSTK                 Throw away return address
1412 04793 6B7D         GOTO   ACCEPT          Jump to process error
1413             *-
1414             *-
1415             * GETDSC
1416             * Entry:  B(XS)=arg count,B(M)=Exad-2
1417             * Exit:   Carry set if too many parms
1418             *         C(S)= parameter descriptor nibble
1419             *         P=0
1420 04797 B25   GETDSC B=B+1  XS              Increment Parm count
1421 0479A 400          RTNC                   Too many parms? Return error
1422 0479D AF9          C=B    W
1423 047A0 109          R1=C                   Save a copy of B in R1
1424 047A3 80D2         P=C    2               Copy current parm count to P
1425 047A7 BF6          CSR    W
1426 047AA BF6          CSR    W
1427 047AD BF6          CSR    W               Get Exad-2
1428 047B0 136          CDOEX
1429 047B3 14A          A=DATO B               Read parm range
1430 047B6 136          CDOEX
1431 047B9 BB0          ASL    X               A(XS)=Max parms, A(1)=Min parms
1432 047BC 9A4          ?B>A   XS              Too many parms?
1433 047BF 00           RTNYES                 Yes, then return error
1434 047C1 FE           C=-C-1 A               Complement Address
1435 047C3 809          C+P+1                  Add Parm number
1436 047C6 FA           C=-C   A               Complement back=(Addr-(P+1)+1)
1437 047C8 20           P=     0
1438 047CA 136          CDOEX
1439 047CD 1564         C=DATO S               Read parameter descriptor
1440 047D1 136          CDOEX
1441 047D4 03           RTNCC                  Return without error
1442             *-
1443             *-
1444 047D6 841   FCNTYP ST=0   String          Assume Numeric function
1445 047D9 3180         LC(2)  NFuncR          Numeric func
1446 047DD 961          ?B=C   B               Is it a numeric func ref?
1447 047E0 00           RTNYES                 Yes, then okay
1448 047E2 309          LC(1)  SFuncR          String func
1449 047E5 965          ?B#C   B               Is it a string func ref?
1450 047E8 9A           GOYES  ParmEr          No, then accept
1451 047EA 851          ST=1   String          Yes, then set string flag
1452 047ED 03           RTNCC
1453             *-
```

```
1454                    EJECT
1455                ****************************************************************
1456                ****************************************************************
1457                **
1458                ** Name:    BOPCOM  -  Compile a binary operator
1459                **
1460                ** Category:   LOCAL
1461                **
1462                ** Purpose:
1463                **      Checks for reducible Binary OPerators on parse stack
1464                **      and COMpiles operation and deletes items from stack
1465                **      if found
1466                **
1467                ** Entry:
1468                **      C(4-0)=THHLL where T is lowest nibble of Look(3),
1469                **      HHLL is range of allowable binary operators.
1470                **
1471                ** Exit:
1472                **      Carry set if nothing found (or compiled)
1473                **      Carry clear if proper <Item> <BOP> <E> <??> found.
1474                **      If found then code will have been compiled and
1475                **      <BOP> and <E> will have been deleted from stack
1476                **
1477                ** Calls:      LOOK2,RANGE,OUT1TK,DELET1
1478                **
1479                ** Uses.......
1480                **  Inclusive: B,C,R1
1481                **
1482                ** Stk lvls:   2
1483                **
1484                ** History:
1485                **
1486                **    Date      Programmer              Modification
1487                **  --------    ----------   ------------------------------------
1488                **  10/04/83    B.S.         Updated documentation
1489                **
1490                ****************************************************************
1491                ****************************************************************
1492 047EF 109    BOPCOM R1=C                    Save parameters
1493 047F2 7AAD        GOSUB  LOOK2
1494 047F6 24          P=     4
1495 047F8 119         C=R1                    Recall parameters
1496 047FB 905         ?B#C   P
1497 047FE 52          GOYES  BOPC10
1498 04800 20          P=     0
1499 04802 AEC         ABEX   B
1500 04805 7ECD        GOSUB  Range
1501 04809 AEC         ABEX   B
1502 0480C 400         RTNC
1503 0480F AEC         ABEX   B
1504 04812 760E        GOSUB  out1tk
1505 04816 AEC         ABEX   B
1506 04819 756D  DL2SXM GOSUB  DELET1
1507 0481D 716D  DL1SXM GOSUB  DELET1
1508 04821 00    RTNSXM RTNSXM
```

```
1509            *-
1510            *-
1511 04823 20    BOPC10 P=     0
1512 04825 02           RTNSC
1513            *-
```

```
1514                    EJECT
1515                ********************************************************************
1516                ********************************************************************
1517                **
1518                ** Name:    CLRPRM  -  Clear parameter count nibble (PRMCNT)
1519                **
1520                ** Category:   PARUTL
1521                **
1522                ** Purpose:
1523                **      Clears parameter count nibble (PRMCNT)
1524                **
1525                ** Entry:
1526                **
1527                ** Exit:
1528                **      P       =  0
1529                **      A(A)    =  0
1530                **      C(A)    =  D0 at time of entry
1531                **
1532                **
1533                **
1534                **
1535                ** Calls:       None
1536                **
1537                ** Uses.......
1538                **  Inclusive: A(A),C(A)
1539                **
1540                ** Stk lvls:   0
1541                **
1542                ** History:
1543                **
1544                **    Date      Programmer              Modification
1545                **  --------    ----------   ----------------------------------
1546                **  10/26/83    B.S.         Added documentation
1547                **
1548                ********************************************************************
1549                ********************************************************************
1550 04827 D0    =CLRPRM A=0     A
1551 04829 136    SETPRM CDOEX
1552 0482C 1B00          DO=(5) =PRMCNT
           000
1553 04833 1580          DAT0=A 1              Write out the nibble
1554 04837 134           D0=C
1555 0483A 01            RTN
1556                *-
1557                *-
```

```
1558                    EJECT
1559                ********************************************************************
1560                ********************************************************************
1561                **
1562                ** Name:    CKLFSD  -  Check if on left hand side of equals
1563                **
1564                ** Category:   LOCAL
1565                **
1566                ** Purpose:
1567                **      Checks for left hand side of equals sign (LET parse)
1568                **
1569                ** Entry:
1570                **      A(B)=Suspected = token.
1571                **      LeftSd(S7) set if looking for left-hand-side
1572                **
1573                ** Exit:
1574                **      Carry clear iff valid left-hand-side has been found
1575                **
1576                ** Calls:      RANGE, LOOK2
1577                **
1578                ** Uses.......
1579                **  Inclusive: C(A),B(B)
1580                **
1581                ** Stk lvls:   1
1582                **
1583                ** Detail:
1584                **      If LeftSd is set then if A(B) is an equals token
1585                **      then the stack is checked for a StartS or StartR.
1586                **      The carry will be returned clear iff one of these
1587                **      is found.
1588                **
1589                ** History:
1590                **
1591                **    Date     Programmer              Modification
1592                **  --------   ----------   ---------------------------------
1593                **  10/05/83   B.S.         Updated documentation
1594                **
1595                ********************************************************************
1596                ********************************************************************
1597 0483C 867    CKLFSD ?ST=0  LeftSd          Looking for left-hand-side?
1598 0483F 00          RTNYES                  No, then return
1599 04841 32A8        LC(3)  t=
           2
1600 04846 936         ?A#C   X                Is it an equals token?
1601 04849 00          RTNYES                  No, then return
1602 0484B 715D        GOSUB  LOOK2            Look at stack
1603 0484F 33C0        LC(4)  (StartS)^(StartR)
           D0
1604 04855 AEC         ABEX   B
1605 04858 7B7D        GOSUB  Range            Set carry if not StartR or StartS
1606 0485C AE4         A=B    B                Restore A(B)
1607 0485F 01          RTN
1608                *-
1609                *-
```

```
1610                        EJECT
1611 04861 D2    DUMARY C=0     A
1612 04863 7C3D         GOSUB  LOOK
1613 04867 7C2F  DUMA05 GOSUB  GETDSC
1614 0486B 491          GOC    DUMAER          Jump if error found
1615 0486E AC5          B=C    S
1616 04871 A46          C=C+C  S
1617 04874 A46          C=C+C  S
1618 04877 A46          C=C+C  S               Is an array required here?
1619 0487A 4E0          GOC    DUMA10          Yes, then look for one
1620 0487D 7000         GOSUB  =NTOKEN         No, then resume expr evaluation
1621 04881 6548         GOTO   EOj
1622             *-
1623             *-
1624 04885 698C  DUMAER GOTO   ACCEPT
1625             *-
1626             *-
1627 04889 119   DUMA10 C=R1
1628 0488C AC9          C=B    S
1629 0488F 109          R1=C                   Save parm desc in R1(S)
1630 04892 7000         GOSUB  =NTOKEN         Get next token
1631 04896 86B          ?ST=0  VarFlg          Is it a variable (or an array)
1632 04899 CE           GOYES  DUMAER          No, then Error:  Array required
1633 0489B 119          C=R1                   Recall updated parm count & desc
1634 0489E AF5          B=C    W               Move function header back into B
1635 048A1 861          ?ST=0  String          Is it a string?
1636 048A4 50           GOYES  DUMA20          No, then skip
1637 048A6 A46          C=C+C  S               Throw away numeric allowed bit
1638 048A9 A46   DUMA20 C=C+C  S
1639 048AC 58D          GONC   DUMAER          Error:  Parameter type
1640 048AF 8E00         GOSUBL =OUTVAR         No, then treat var as an array
          00
1641 048B5 D9           C=B    A               Copy parameter count back
1642 048B7 71DC         GOSUB  INSRT1          Update parameter count in stack
1643 048BB 8E00         GOSUBL =GNXTCR         Get next character
          00
1644                                           (should be comma or ")")
1645 048C1 171          D1=D1+ 2               Skip past this character
1646 048C4 31C2         LCASC  \,\
1647 048C8 966          ?A#C   B               Is it a comma?
1648 048CB 60           GOYES  DUMA70          No, then look for right paren
1649 048CD 699F         GOTO   DUMA05          Yes, then check if another dummy
1650                                           array needed
1651             *-
1652             *-
1653 048D1 8E00  DUMA70 GOSUBL =DUMA80         Check min number of parameters
          00
1654 048D7 4DA          GOC    DUMAER          Error if not ")" or
1655                                           if minimum not met
1656 048DA 74AC         GOSUB  DELET1          Delete an item from stack
1657 048DE 74FE         GOSUB  FCNTYP          Set status bit for type of fcn
1658 048E2 6E5A         GOTO   FCNCOM          Compile function
1659 048E6              END
```

```
=ACCEPT  Abs   17679 #0450F -   865   356   444   607   715  1412  1624
 ACPT10  Abs   17744 #04550 -   887   882
 ACPT20  Abs   17768 #04568 -   896   885
 ACPT30  Abs   17780 #04574 -   899   889
 ACPT40  Abs   17784 #04578 -   900   892   895
=ARRANGE Abs   17873 #045D1 -  1112   291
 AVMEME  Ext                -   870
 AVMEMS  Ext                -  1327  1341
 Accept  Abs   17447 #04427 -   715   692   720   723
 ArgLst  Abs       2 #00002 -   266   370
 BIG     Abs   18108 #046BC -  1287  1251
 BOPC10  Abs   18467 #04823 -  1511  1497
 BOPCOM  Abs   18415 #047EF -  1492   487   522   549   570   579   811   817
 CHKC40  Abs   18159 #046EF -  1305  1303
 CHKCON  Abs   18003 #04653 -  1246   301
 CKLFSD  Abs   18492 #0483C -  1597   564   824
=CLRPRM  Abs   18471 #04827 -  1550
=CMPB#C  Abs   17854 #045BE -  1078   509
 CMPLX   Ext                -   699
 COLLIS  Abs   17924 #04604 -  1187  1178  1299
 CON1    Abs   17103 #042CF -   575   572
 CONC10  Abs   18059 #0468B -  1264  1255
 CONC20  Abs   17938 #04612 -  1224  1263
=CONCOM  Abs   18046 #0467E -  1261  1254
 CONWRN  Abs   18171 #046FB -  1310  1284
 CSRW3   Ext                -   391
 Conjun  Abs       6 #00006 -   254   569   573
 D=AVME  Ext                -   276  1340
 DELET1  Abs   17794 #04582 -   942   388   657   708   731   941  1506  1507
                                1656
 DELET2  Abs   17790 #0457E -   941   613   676   693   729
 DL1SXM  Abs   18461 #0481D -  1507   511   537   628
 DL2SXM  Abs   18457 #04819 -  1506   598
 DRANGE  Ext                -  1224
 DUMA05  Abs   18535 #04867 -  1613   711  1649
 DUMA10  Abs   18569 #04889 -  1627  1619
 DUMA20  Abs   18601 #048A9 -  1638  1636
 DUMA70  Abs   18641 #048D1 -  1653  1648
 DUMA80  Ext                -  1653
 DUMAER  Abs   18565 #04885 -  1624  1614  1632  1639  1654
 DUMARY  Abs   18529 #04861 -  1611   395
 E0      Abs   16374 #03FF6 -   286   353
 E0+05   Abs   16419 #04023 -   296   292
 E0+10   Abs   16429 #0402D -   300   295
 E0+20   Abs   16443 #0403B -   307   302
 E0+30   Abs   16462 #0404E -   314   309
 E0+40   Abs   16488 #04068 -   322   315
 E0+40.  Abs   16529 #04091 -   339   331
 E0+41   Abs   16587 #040CB -   356   317   341   349
 E0+41.  Abs   16591 #040CF -   359   324
 E0+42   Abs   16600 #040D8 -   362   327   360
 E0+44   Abs   16602 #040DA -   363   345
 E0+46   Abs   16638 #040FE -   376   374
 E0+48   Abs   16696 #04138 -   395   387   413
 E0+52   Abs   16700 #0413C -   398   371
```

```
 E0+54    Abs   16709 #04145 -   401   392
 E0+60    Abs   16724 #04154 -   408   385
 E0+64    Abs   16763 #0417B -   421   410
 E0+68    Abs   16829 #041BD -   444   415   419   431   510
 E0-10    Abs   16370 #03FF2 -   283   290   496   566   807
 E0-J0    Abs   17077 #042B5 -   566   529   544   547   577   587
 E0-J2    Abs   17614 #044CE -   807   815   825
 E0j      Abs   16583 #040C7 -   353  1621
 E1       Abs   17131 #042EB -   584
 E1+03    Abs   17147 #042FB -   588   826
 E1+03j   Abs   17671 #04507 -   826   823
 E1+05    Abs   17160 #04308 -   594   590
 E1+08    Abs   17186 #04322 -   604   600
 E1+09    Abs   17190 #04326 -   607   618   636
 E1+10    Abs   17194 #0432A -   610   597
 E1+15    Abs   17198 #0432E -   612
 E1+20    Abs   17209 #04339 -   615   405
 E1+30    Abs   17355 #043CB -   681   591
 E1+32    Abs   17429 #04415 -   707   689
 E1+35    Abs   17451 #0442B -   718   684
 E1-10    Abs   17123 #042E3 -   582
 E2       Abs   17355 #043CB -   680
=EXPP10   Abs   16355 #03FE3 -   277
=EXPPAR   Abs   16345 #03FD9 -   275
=EXPPLS   Abs   16348 #03FDC -   276
 F1       Abs   16862 #041DE -   492   489
 F1+05    Abs   16875 #041EB -   499   495
 F1+10    Abs   16902 #04206 -   506   520
 F1+20    Abs   16927 #0421F -   517   503
 FCNC05   Abs   17251 #04363 -   638   634
 FCNC10   Abs   17270 #04376 -   645   643
 FCNC15   Abs   17272 #04378 -   646   652
 FCNC20   Abs   17281 #04381 -   650   632
 FCNC30   Abs   17306 #0439A -   658   677
 FCNC32   Abs   17319 #043A7 -   662   659
 FCNC35   Abs   17328 #043B0 -   667   663
 FCNC40   Abs   17332 #043B4 -   670   655
 FCNC42   Abs   17338 #043BA -   672   647
 FCNC45   Abs   17348 #043C4 -   676
 FCNCOM   Abs   17217 #04341 -   619  1658
 FCNTYP   Abs   18390 #047D6 -  1444   610  1657
 FLTCOM   Abs   18032 #04670 -  1255  1279
=FRange   Abs   46186 #0B46A -   263   342
 Factor   Abs       2 #00002 -   250   486   490
 GETDSC   Abs   18327 #04797 -  1420  1404  1613
 GNXCR+   Ext                -   380   416
 GNXTCR   Ext                -  1643
 HNDL20   Ext                -  1316
 INSRT1   Abs   17804 #0458C -   981   491   513   526   541   560   574   583
                             -   710   801  1642
 InvalE   Abs       0 #00000 -   264   277   439   896
 LASTFN   Abs     180 #000B4 -   262   263
 LOOK     Abs   17827 #045A3 -  1039   382   425  1612
 LOOK2    Abs   17824 #045A0 -  1038   499   530   594   627   681   706   730
                             -  1493  1602
```

```
 LeftSd  Abs       7 #00007 -   268   275  1597
 MEMBER  Ext                -   289
 MEMERR  Ext                -  1187
 MFWRQ8  Ext                -  1338
 MemErr  Abs   18144 #046E0 -  1299  1326
 NExpr   Abs       7 #00007 -   255   578   582   880
 NFuncR  Abs       8 #00008 -   256   372   686   697  1445
 NPrim   Abs       0 #00000 -   248
 NTOKEN  Ext                -   438  1189  1620  1630
 NumExp  Abs       3 #00003 -   267   440   584   599   820   865   886   897
                              1406
 OUT1TK  Ext                -  1226
 OUTBYj  Abs   16341 #03FD5 -   272   535   661  1271  1305
 OUTLI1  Ext                -   316
 OUTNBC  Ext                -   557   728
 OUTNBS  Ext                -  1258
 OUTNIB  Ext                -   437
 OUTVAR  Ext                -   296   310   435   646  1640
 OUTbyt  Ext                -   272
 OVFL:1  Abs   18155 #046EB -  1304  1293
 OVFL:2  Abs   18148 #046E4 -  1302  1295
 OVFNIB  Ext                -  1288
 OVP     Ext                -  1307
 P1      Abs   16840 #041C8 -   486   514
=P1-10   Abs   16833 #041C1 -   484   297   601   664  1349
 P1-10j  Abs   16425 #04029 -   297   304
 P1-J2   Abs   17182 #0431E -   601
 PARM10  Abs   18315 #0478B -  1409  1407
 PARMCK  Abs   18300 #0477C -  1404   612   707   724
 PRESCN  Ext                -   352
 PRMCNT  Ext                -  1552
=PUSH-3  Abs   17886 #045DE -  1172   399  1188
=PUSH-P  Abs   17888 #045E0 -  1173   363   379   704
 ParmEr  Abs   18321 #04791 -  1411  1243  1296  1405  1450
 QUOTCK  Ext                -   314
 R<RST2  Ext                -  1336
 RANGE   Ext                -  1113
 REL+10  Abs   17081 #042B9 -   569   563   565
 REL-05  Abs   17053 #0429D -   559   551
 REL1    Abs   17061 #042A5 -   561
 RESPTR  Ext                -   350   898
 RST2<R  Ext                -  1339
 RTNSXM  Abs   18465 #04821 -  1508   303   318   899
=Range   Abs   17879 #045D7 -  1113   294   323   340   344   502   528   586
                               687  1253  1262  1500  1605
 Relat   Abs       5 #00005 -   253   548   559
 SCAN    Abs   17930 #0460A -  1188   283   485   775
 SE1     Abs   17522 #04472 -   776
 SE1+05  Abs   17570 #044A2 -   793   786   791
 SE1+10  Abs   17587 #044B3 -   799   789
 SE1+20  Abs   17618 #044D2 -   810   778
=SE1-10  Abs   17512 #04468 -   773   311   319   604   667   734
 SETPRM  Abs   18473 #04829 -  1551   361
 SExpr   Abs       1 #00001 -   249   810   816   883
 SFuncR  Abs       9 #00009 -   257   375  1448
```

SMALL	Abs	17954	#04622	-	1229	1248					
SUB$	Ext			-	799	803					
SUM+10	Abs	17037	#0428D	-	553	829					
SUM+j	Abs	17675	#0450B	-	829	819					
SUM-10	Abs	17002	#0426A	-	541	533					
SUM1	Abs	17006	#0426E	-	542	538					
StartR	Abs	12	#0000C	-	259	281	428	640	787	890	1603
StartS	Abs	13	#0000D	-	260	732	784	893	1603		
StartV	Abs	14	#0000E	-	261	887					
String	Abs	1	#00001	-	265	373	414	633	662	1444	1451	1635
SubstR	Abs	10	#0000A	-	258	686	721	805			
Sum	Abs	4	#00004	-	252	531					
T1	Abs	16958	#0423E	-	527						
Term	Abs	3	#00003	-	251	521	525				
UNDF10	Abs	18097	#046B1	-	1280	1272					
UNDF:1	Abs	18072	#04698	-	1270	1240	1278				
UNDF:2	Abs	18083	#046A3	-	1275	1242					
UNFNIB	Ext			-	1230						
UNP	Ext			-	1281						
VarFlg	Abs	11	#0000B	-	269	384	398	614	658	1631	
eOVFLW	Ext			-	1306						
eUNFLW	Ext			-	1280						
outitk	Abs	17948	#0461C	-	1226	504	656	1504			
outnbs	Abs	18040	#04678	-	1258	333	671	675	1267		
t&	Abs	137	#00089	-	262	810	810	813			
t*	Abs	131	#00083	-	262	521	527				
t+	Abs	135	#00087	-	262	517	534	542			
t-	Abs	130	#00082	-	262	500	545				
t=	Abs	650	#0028A	-	270	1599					
tADIGO	Abs	96	#00060	-	262	293					
tADIG9	Abs	105	#00069	-	262	293					
tAND	Abs	139	#0008B	-	262	569	569	575			
tARRAY	Abs	125	#0007D	-	262	322	630				
tCMPLX	Abs	122	#0007A	-	262	700					
tCOMMA	Abs	241	#000F1	-	262	682					
tDIV	Abs	134	#00086	-	262	521	527				
tEXOR	Abs	140	#0008C	-	262	578	585				
tFLT1	Abs	29	#0001D	-	262	1252					
tFLT12	Abs	18	#00012	-	262	1252					
tFN	Abs	124	#0007C	-	262	322	650				
tINF	Abs	112	#00070	-	262	1302					
tINT12	Abs	2	#00002	-	262	1261					
tINT2	Abs	12	#0000C	-	262	1261					
tISUB$	Abs	167	#000A7	-	262	726					
tLPRP	Abs	170	#000AA	-	262	660					
tMAXRL	Abs	108	#0006C	-	262	1304					
tNOT	Abs	129	#00081	-	262	500					
tOR	Abs	141	#0008D	-	262	578	585				
tRELOP	Abs	138	#0008A	-	262	270	548	548	561	816	816	821
tSMALL	Abs	17	#00011	-	262	1246					
tSVAR	Abs	45	#0002D	-	262	307					
tXFN	Abs	179	#000B3	-	262	325	653				
tXWORD	Abs	239	#000EF	-	262	347					
t^	Abs	128	#00080	-	262	486	486	493	508		

Input Parameters

  Source file name is SB&EXP::MS

  Listing file name is SB/EXP:TI:ML::-1

  Object file name is SB%EXP:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *      A    BBBB    &     L      EEEEE  X   X
 2            *     A A   B   B  & &    L      E      X   X
 3            *    A   A  B   B  & &    L      E       X X
 4            *    A   A  BBBB    &     L      EEEE     X
 5            *    AAAAA  B   B  & & &  L      E       X X
 6            *    A   A  B   B  &  &   L      E      X   X
 7            *    A   A  BBBB    && &  LLLLL  EEEEE  X   X
 8                      TITLE  Lexical Analyzer <831212.1512>
 9 048E6                ABS    #048E6
10            ********************************************************************
11            ********************************************************************
12            **
13            ** Name:(S) NTOKNL  -  Lex Analysis
14            ** Name:(S) NTOKEN  -  Lex Analysis
15            ** Name:(S) PRESCN  -  Lex Analysis
16            ** Name:(S) RESCAN  -  Lex Analysis
17            ** Name:(S) VRIABL  -  Lex Analysis
18            ** Name:    SHFTKN  -  Lex Analysis
19            ** Name:(S) ALLDUN  -  Lex Analysis
20            ** Name:    HOWARD  -  Lex Analysis
21            ** Name:(S) LEAVE   -  Lexical Analysis
22            **
23            ** Category:   PARUTL
24            **
25            ** Purpose:
26            **      The lexical analyzer scans strings of ASCII characters
27            **      and associates unique numbers (tokens) with particular
28            **      substrings (lexemes).  The tokens are used by language
29            **      parsing routines and interpreters.
30            **
31            ** Entry:
32            **      Many different entry points for different purposes.
33            **
34            **      NTOKNL - Looks for line number, or any other lexeme.
35            **      NTOKEN - Looks for any lexeme not a line number.
36            **
37            **               D1 is current input buffer position.
38            **               DO is current output buffer position.
39            **               D(A) is end of output buffer.
40            **
41            **      PRESCN - Same as RESCAN, except output pointer is still
42            **               in DO, instead of C(A).
43            **
44            **      RESCAN - Looks for another token corresponding to a
45            **               lexeme.
46            **
47            **               IMPORTANT ENTRY POINT.  There is where the
48            **               lexical analyzer can be restarted if an
49            **               undesired match occurred with an XWORD.
50            **
51            **               D1 is reset to start of lexeme to be
52            **                 rescanned.
53            **               A(A) is Lexbuffer pointer returned for token
54            **                 to be replaced.
55            **               C(A) is reset to start of token to be
```

```
56          **                replaced.
57          **              This is the output pointer, which will be in
58          **              D0 upon exit. This pointer is not actually
59          **              used by this routine.
60          **              D(A) is end of output buffer.
61          **
62          **      VRIABL - Looks for Basic variable name.
63          **
64          **              D1 is current input buffer position.
65          **              R0 is end of output buffer (done by previous
66          **              entry points).
67          **
68          **      SHFTKN - Places token in C(B) in front of tokens in A.
69          **
70          **              D1 is new input buffer position.
71          **              D0 is lexbuffer pointer.
72          **              D(A) is execaddress, if there is one.
73          **              R0 is end of output buffer.
74          **
75          **      ALLDUN - Restores output buffer pointer to D0.
76          **
77          **              D1 is new input buffer position.
78          **              D0 is lexbuffer pointer.
79          **              D(A) is execaddress, if there is one.
80          **              R0 is end of output buffer.
81          **
82          **      HOWARD - Restores output buffer pointer to D0.
83          **
84          **              D1 is new input buffer position.
85          **              C(A) is lexbuffer pointer.
86          **              D(A) is execaddress, if there is one.
87          **              R0 is end of output buffer.
88          **
89          **      LEAVE  - Restores end of output buffer to D(A).
90          **
91          **              D1 is new input buffer position.
92          **              D0 is current output buffer position.
93          **              D(A) is end of output buffer.
94          **
95          ** Exit:
96          **      P=0.
97          **      D1 is new input pointer.
98          **      D0 is current output pointer.
99          **      A contains token, up to 14 nibbles in length.
100         **      B[A] is execution address (if there is one).
101         **      B[X] is numeric constant exponent, if there is one.
102         **      C[S] is lexbuffer pointer used for RESCAN.
103         **      D[A] is end of output buffer.
104         **
105         ** Calls:      ARGCHK, BLDVAR, D=WORD, DGTSTR, GNXTCR, IOFNDO,
106         **             LDZERO, NUMSCN, Range, SCAN, STLXPT, STRCHK.
107         **
108         ** Uses.......
109         **             A,B,C,P,D0,D1,R0,S0-S3,S11.
110         **
```

```
111             ** Stk lvls:   2
112             **
113             ** Detail:
114             **      The lexical analyzer consists of two parts: scanner
115             **      and lexicon.  The scanner is the code described here
116             **      with several entry points, one major subroutine
117             **      (NUMSCN) and many smaller subroutines.
118             **
119             **      The lexicon is a set of tables:
120             **
121             **        LXTYPT (lexical type table) is a table of character
122             **        categories, or types, which lives in system ROM.
123             **        This table helps the scanner reduce the time
124             **        selecting which scanning method to use:
125             **          Type 0 - Direct: Use transfer character in type
126             **                           table as token.
127             **          Type 1 - Word: Scan text table for string match.
128             **          Type 2 - Relational: Scan for relational operator.
129             **          Type 3 - Number: Call NUMSCN to format constant.
130             **
131             **        LEXBFR (lexfile buffer) is an I/O buffer in system
132             **        RAM which contains lextable IDs and maintable
133             **        addresses.
134             **
135             **        LXSPDT (speed table) is an optional table within
136             **        each lexfile which tells where in text table lexemes
137             **        with a particular first character begin.
138             **
139             **        LXTXTT (text table) is a table in every lexfile
140             **        containing the following text information:
141             **          Lexeme length - 1 nibble,
142             **          Lexeme text - 2-16 nibbles,
143             **          Lexeme token - 2 nibbles.
144             **
145             **        MAINT (main table) is a table in every lexfile which
146             **        contains token information:
147             **          Text offset - 3 nibbles.
148             **                        Locates text in text table; used in
149             **                        decompiling.
150             **          Execaddress - 5 nibbles.
151             **                        Self-relative pointer to token's
152             **                        execution address.
153             **          Characterization - 1 nibble.
154             **                             Syntactic class and spacing
155             **                             information.
156             **
157             ** History:
158             **
159             **    Date     Programmer             Modification
160             **  --------   ----------   ----------------------------------
161             **  04/01/83   SA           Figured out register & subr usage
162             **  10/17/83   NM           Attempted to document
163             **
164             ****************************************************************
165             ****************************************************************
```

```
166 048E6 841   =NTOKNL ST=0   1            INITIALIZE
167 048E9 842            ST=0   2
168 048EC D0             A=0    A
169 048EE DB             C=D    A            SAVE END-OF-BUFFER-ADDRESS IN B
170 048F0 D5             B=C    A
171 048F2 8E00           GOSUBL =GNXTCR      SKIP LEADING BLANKS
          00
172 048F8 7A04           GOSUB  STLXPT       Store Lex Pointer
173 048FC 305            LCHEX  5
174 048FF 816            CSRC
175 04902 7916           GOSUB  LDZERO       SKIP LEADING ZEROES
176 04906 D9             C=B    A            RESTORE END-OF-BUFFER-ADDRESS
177 04908 D7             D=C    A
178 0490A D1             B=0    A
179 0490C CD             B=B-1  A
180 0490E E5    LLOOP    B=B+1  A            INCREMENT NORMALIZER
181 04910 7916           GOSUB  DGTSTR       READ CHARACTER
182 04914 59F            GONC   LLOOP        IF DIGIT, REPEAT LLOOP
183 04917 861            ?ST=0  1            NO DIGITS AT ALL?
184 0491A 12             GOYES  NTOKEN       IF NOT, GOTO NTOKEN
185 0491C 862            ?ST=0  2            NOTHING BUT ZEROES?
186 0491F 01             GOYES  L#0000       IF SO, GOTO L#0000
187 04921 94A            ?C=0   S            TOO MANY DIGITS FOR LINE NUMBER
188 04924 D0             GOYES  RESTRT       IF SO, GOTO RESTRT
189 04926 31F0           LCHEX  0F
190 0492A AEA            A=C    B            PLACE LINE NUMBER TOKEN
191 0492D 01             RTN                 RETURN
192 0492F E5    L#0000   B=B+1  A            INCREMENT NORMALIZER
193 04931 C5    RESTRT   B=B+B  A            BACK UP TEXT POINTER
194 04933 137            CD1EX
195 04936 E9             C=C-B  A
196 04938 135            D1=C
197 0493B 8E00  =NTOKEN  GOSUBL =GNXTCR      SKIP LEADING BLANKS
          00
198 04941 841            ST=0   1            INITIALIZE
199 04944 842            ST=0   2
200 04947 843            ST=0   3
201 0494A 84B            ST=0   11
202 0494D D0             A=0    A
203 0494F D1             B=0    A
204 04951 71B3           GOSUB  STLXPT       Store Lex Pointer
205 04955 14B            A=DAT1 B            READ CHARACTER
206 04958 31D0           LCHEX  0D
207 0495C 966            ?A#C   B            CARRIAGE RETURN?
208 0495F D0             GOYES  NOT-CR       IF NOT, GOTO NOT-CR
209 04961 3100           LC(2)  =tEOL        LOAD EOL TOKEN
210 04965 AEA   AWFUL    A=C    B
211 04968 6870           GOTO   EXIT1        GOTO EXIT1
212 0496C 3312  NOT-CR   LCASC  \^!\
          E5
213 04972 7000           GOSUB  =Range       CAN PROGRAM OPERATE ON CHARACTER?
214 04976 5A1            GONC   OKCHAR       IF SO, GOTO OKCHAR
215 04979 31FD           LCHEX  DF
216 0497D 0E66           A=A&C  B            CONVERT TO UPPERCASE
217 04981 3314           LCASC  \ZA\
```

```
          A5
218 04987 7000        GOSUB  =Range          LETTER?
219 0498B AE2         C=0    B              LOAD NULL IN CASE OF ERROR
220 0498E 46D         GOC    AWFUL          IF NOT, GOTO AWFUL
221 04991 AE8  OKCHAR B=A    B              COMPUTE LEXTYPE TABLE ENTRY
222 04994 A38         B=B+A  X
223 04997 A38         B=B+A  X
224 0499A 3400        LC(5)  (=LXTYPT)-99   TYPE TABLE MINUS 99
          000
225 049A1 C9          C=C+B  A
226 049A3 136         CDOEX
227 049A6 1523        A=DATO X              READ LEXTYPE AND TRANSFER CHAR
228 049AA 134         DO=C
229 049AD A2C  MISC   A=A-1  XS             MISCELLANEOUS STUFF?
230 049B0 5A0         GONC   NUMBER         IF NOT, GOTO NUMBER
231 049B3 840         ST=0   0              CLEAR BEGIN BASIC FLAG
232 049B6 171         D1=D1+ 2              POINT TO NEXT CHARACTER
233 049B9 03          RTNCC                 RETURN CARRY CLEAR
234 049BB A2C  NUMBER A=A-1  XS             LETTER?
235 049BE 496         GOC    WORD           IF SO, GOTO WORD
236 049C1 870         ?ST=1  0              CHARACTERIZATION REQUEST?
237 049C4 D1          GOYES  EXIT1          IF SO, GOTO EXIT1
238 049C6 928         ?A=0   XS             RELATIONAL OPERATOR?
239 049C9 93          GOYES  RELOP          IF SO, GOTO RELOP
240 049CB DB          C=D    A
241 049CD 108         R0=C                  SAVE END-OF-BUFFER IN R0
242 049D0 7443        GOSUB  NUMSCN         COMPILE CONSTANT
243 049D4 118         C=R0
244 049D7 D7          D=C    A              RESTORE END-OF-BUFFER TO D
245 049D9 04          SETHEX                RESET STATE
246 049DB 842         ST=0   2
247 049DE 841         ST=0   1
248 049E1 840  EXIT1  ST=0   0
249 049E4 03          RTNCC                 RETURN CARRY CLEAR
250 049E6 A86  RELOP1 C=A    P
251 049E9 80D0        P=C    0
252 049ED 3600        LCHEX  1248000        LOAD PREDICATE BIT IN XS-FIELD
          0842
          1
253 049F6 20          P=     0
254 049F8 0E2E RELOP2 A=A!C  XS             INSTALL BIT IN TOKEN
255 049FC 171         D1=D1+ 2              INCREMENT TEXT POINTER
256 049FF 14B         A=DAT1 B              READ NEXT CHARACTER
257 04A02 3232 RELOP  LCHEX  D23
          D
258 04A07 962         ?A=C   B              HATCHMARK?
259 04A0A EE          GOYES  RELOP2         IF SO, REPEAT RELOP2
260 04A0C 33C3        LCASC  \?<\
          F3
261 04A12 7000        GOSUB  =Range         ANOTHER RELATIONAL CHARACTER?
262 04A16 5FC         GONC   RELOP1         IF SO, REPEAT RELOP
263 04A19 3100        LC(2)  =tRELOP        LOAD RELOP TOKEN
264 04A1D AEA         A=C    B
265 04A20 03          RTNCC                 RETURN CARRY CLEAR
266 04A22 8C00 BFRERR GOLONG =CORUPT
```

```
          00
    267 04A28 137   WORD   CD1EX
    268 04A2B D5           B=C    A              SAVE INPUT POINTER
    269 04A2D 3200         LC(3)  =bLEX
          0
    270 04A32 8F00         GOSBVL =IOFNDO
          000
    271 04A39 58E          GONC   BFRERR
    272 04A3C 8A8          ?A=0   A              BUFFER EMPTY?
    273 04A3F 3E           GOYES  BFRERR
    274 04A41 1CA          D1=D1- 11             PT D1 11 NIBS PRIOR TO 1ST ADDR
    275 04A44 D4           A=B    A              INPUT PTR
    276 04A46 133          AD1EX                 RESTORE INPUT PTR
    277 04A49 136 =PRESCN  CDOEX                 SAVE OUTPUT PTR
    278 04A4C 1B00 =RESCAN DO=(5) =FUNCDO        STORE OUTPUT POINTER
          000
    279 04A53 144          DATO=C A
    280 04A56 130          DO=A                  POINT AT LEX BUFFER
    281 04A59 DB           C=D    A              SAVE END-OF-BUFFER IN R0
    282 04A5B 108          R0=C
    283 04A5E 7CA1         GOSUB  D=WORD         READ INPUT & CONVERT CASE
    284 04A62 D2           C=0    A              COMPUTE SPEED TABLE OFFSET
    285 04A64 AEB          C=D    B
    286 04A67 D5           B=C    A
    287 04A69 A65          B=B+B  B
    288 04A6C A31          B=B+C  X
    289 04A6F 3200         LC(3)  =oSPDTB        OFFSET FOR SPEED TABLE CALC
          0
    290 04A74 ED           B=C-B  A
    291 04A76 6910         GOTO   LEX            GOTO LEX
    292 04A7A 6941 VBLLNK  GOTO   VRIABL         SHORT JUMP LINKAGE
    293 04A7E 07   NXTROM  C=RSTK                POP LEXBUFFER POINTER
    294 04A80 134          DO=C
    295 04A83 185          DO=DO- 6
    296 04A86 14A          A=DATO B              READ ROM NUMBER
    297 04A89 968          ?A=0   B              TABLES EXHAUSTED?
    298 04A8C EE           GOYES  VBLLNK         IF SO, GOTO VRIABL
    299 04A8E 20           P=     0
    300 04A90 16A  LEX     DO=DO+ 11             MOVE LEXBUFFER PTR TO NEXT ADDR
    301 04A93 165          DO=DO+ 6
    302 04A96 146          C=DATO A              READ ROM TABLE ADDRESS
    303 04A99 136          CDOEX
    304 04A9C 06           RSTK=C                PUSH LEXBUFFER PTR ON HARDWR STK
    305 04A9E 180          DO=DO- =oSPDn2
    306 04AA1 146          C=DATO A              READ LEXTABLE RELATIVE ADDRESS
    307 04AA4 A06          C=C+C  P              SPEED TABLE PRESENT?
    308 04AA7 F6           CSR    A
    309 04AA9 132          ADOEX
    310 04AAC 4F0          GOC    SCNADR         IF NOT, GOTO SCNADR
    311 04AAF C2           C=A+C  A              COMPUTE TEXT TABLE ABS ADDRESS
    312 04AB1 E0           A=A-B  A              COMPUTE SPEED TABLE ENTRY
    313 04AB3 130          DO=A
    314 04AB6 D0           A=0    A
    315 04AB8 1523         A=DATO X              READ SPEED TABLE ENTRY
    316 04ABC CA   SCNADR  A=A+C  A              COMPUTE TEXT TABLE ENTRY POINTER
```

```
    317 04ABE 130          DO=A
    318 04AC1 7B71         GOSUB  SCAN          SCAN ROM TABLE
    319 04AC5 48B          GOC    NXTROM        IF NOT FOUND, REPEAT NXTROM
    320 04AC8 07           C=RSTK
    321 04ACA AE8   GARRR  B=A    B
    322 04ACD 134          DO=C
    323 04AD0 183          DO=DO- 4
    324 04AD3 14E          C=DATO B             READ TOKEN BASE
    325 04AD6 B6A          A=A-C  B             NORMALIZE TOKEN
    326 04AD9 163          DO=DO+ 4
    327 04ADC 146          C=DATO A             READ MAIN TABLE BASE ADDRESS
    328 04ADF C2           C=A+C  A             COMPUTE MAIN TABLE POINTER
    329 04AE1 C4           A=A+A  A
    330 04AE3 C4           A=A+A  A
    331 04AE5 C4           A=A+A  A
    332 04AE7 C2           C=A+C  A
    333 04AE9 185          DO=DO- 6             MOVE LEXBUFFER PTR TO ROM NUMBER
    334 04AEC 136          CDOEX
    335 04AEF D7           D=C    A
    336 04AF1 162          DO=DO+ 3
    337 04AF4 15E5         C=DATO 6             READ CHARACTER'ZN & EXECOFFSET
    338 04AF8 132          ADOEX                COMPUTE EXECADDRESS
    339 04AFB C2           C=A+C  A
    340 04AFD 132          ADOEX
    341 04B00 80D5         P=C    5
    342 04B04 969          ?B=0   B             WEIRD TOKEN?
    343 04B07 32           GOYES  WEIRD         IF SO, GOTO WEIRD
    344 04B09 DF           CDEX   A
    345 04B0B 134          DO=C
    346 04B0E 89F          ?P=    15            FUNCTION?
    347 04B11 A7           GOYES  FCN           IF SO, GOTO FCN
    348 04B13 860          ?ST=0  0             REQUEST FOR CHARACTERIZATION?
    349 04B16 A0           GOYES  SETUP         IF NOT, GOTO SETUP
    350 04B18 0B           CSTEX                SWAP IN CHARACTERIZATION BITS
    351 04B1A 80F0         CPEX   0
    352 04B1E 0B           CSTEX
    353 04B20 20    SETUP  P=     0
    354 04B22 3100         LC(2)  =tXWORD       LOAD XWORD TOKEN
    355 04B26 6380         GOTO   ROMNUM        GOTO ROMNUM
    356 04B2A 06    WEIRD  RSTK=C
    357 04B2C 03           RTNCC                GOTO SECONDARY LEX ANALYZER...
    358 04B2E 5B9   GARRRj GONC   GARRR         Inrange jump from below
    359 04B31 D2    =FN-GO C=0    A
    360 04B33 306          LCHEX  6
    361 04B36 C3           D=C+D  A             FIDDLE WITH LEXBUFFER POINTER
    362 04B38 1C3          D1=D1- 4             BACK UP 2 BYTES
    363 04B3B 143          A=DAT1 A             READ 2 BYTES
    364 04B3E 173          D1=D1+ 4
    365 04B41 B24          A=A+1  XS            "GO"?
    366 04B44 5B2          GONC   FNSCAN        IF NOT, GOTO FNSCAN
    367 04B47 8E00         GOSUBL =GNXTCR
              00
    368 04B4D DB           C=D    A
    369 04B4F D5           B=C    A             MOVE LEXBUFFER POINTER
    370 04B51 79B0         GOSUB  D=WORD        READ UPPERCASE WORD
```

```
371 04B55 1B27          DO=(5) =FUNNY
          C40
372 04B5C 70E0          GOSUB  SCAN          SCAN TABLE
373 04B60 D9            C=B    A             MOVE LEXBUFFER POINTER BACK
374 04B62 5BC           GONC   GARRRj        IF FOUND, REPEAT GARRR
375 04B65 D7            D=C    A
376 04B67 D0            A=0    A             CREATE NULL TOKEN
377 04B69 840           ST=0   0
378 04B6C 6780          GOTO   CHARLY        GOTO CHARLY
379
380 04B70 7111  FNSCAN  GOSUB  BLDVAR        CONSTRUCT VARIABLE TOKEN
381 04B74 4C4           GOC    FNERR         IF NO VARIABLE, GOTO FNERR
382 04B77 AF4           A=B    W
383 04B7A 3400          LC(5)  =FN           LOAD USERFUNCTION EXECADDRESS
          000
384 04B81 D7            D=C    A
385 04B83 3100          LC(2)  =tFN          LOAD FN TOKEN
386 04B87 6E50          GOTO   SHFTKN        GOTO SHFTKN
387
388 04B8B 20    FCN     P=     0
389 04B8D 870           ?ST=1  0             CHARACTERIZATION REQUEST?
390 04B90 31            GOYES  EXIT2         IF SO, GOTO EXIT2
391 04B92 1C1           D1=D1- 2
392 04B95 14B           A=DAT1 B             REREAD LAST CHARACTER
393 04B98 171           D1=D1+ 2
394 04B9B 7951          GOSUB  STRCHK        CHECK FOR STRING FUNCTION
395 04B9F 7141          GOSUB  ARGCHK        CHECK FOR ARGUMENT LIST
396 04BA3 840   EXIT2   ST=0   0             CLEAR CHARACTERIZATION REQUEST
397 04BA6 3100          LC(2)  =tXFN         LOAD XFN TOKEN
398 04BAA 14A   ROMNUM  A=DAT0 B             READ ROM NUMBER
399 04BAD AEC           ABEX   B
400 04BB0 969           ?B=0   B             MAINFRAME LEXEME?
401 04BB3 C3            GOYES  ALLDUN        IF SO, GOTO ALLDUN
402 04BB5 BB0           ASL    X             SHIFT LOCAL TOKEN LEFT
403 04BB8 F0            ASL    A
404 04BBA AE4           A=B    B             PLACE ROM NUMBER
405 04BBD 6820          GOTO   SHFTKN        GOTO SHFTKN
406 04BC1 1C3   FNERR   D1=D1- 4             BACK UP TO "F"
407 04BC4 7DB0 =VRIABL GOSUB  BLDVAR        CONSTRUCT VARIABLE TOKEN
408 04BC8 AE0           A=0    B             LOAD NULL IN CASE OF ERROR
409 04BCB 432           GOC    ALLDUN        IF NO VARIABLE, GOTO ALLDUN
410 04BCE AF4           A=B    W
411 04BD1 85B           ST=1   11            RECORD VARIABLE
412 04BD4 862           ?ST=0  2             ARRAY?
413 04BD7 81            GOYES  ALLDUN        IF NOT, GOTO ALLDUN
414 04BD9 3400          LC(5)  =ARRAY        LOAD ARRAY EXECADDRESS
          000
415 04BE0 D7            D=C    A
416 04BE2 3100          LC(2)  =tARRAY       LOAD ARRAY TOKEN
417 04BE6 BF0   =SHFTKN ASL    W             SHIFT EXISTING TOKENS LEFT
418 04BE9 BF0           ASL    W
419 04BEC AEA           A=C    B             PLACE HEADER TOKEN
420 04BEF 136   =ALLDUN CDOEX               RESTORE OUTPUT POINTER TO DO
421 04BF2 D5    =HOWARD B=C    A
422 04BF4 1B00  CHARLY  DO=(5) =FUNCDO
```

```
           000
423 04BFB 146          C=DAT0 A
424 04BFE 134          DO=C
425 04C01 118  =LEAVE  C=RO                  LEAVE TOKENS IN A
426 04C04 DF           CDEX   A              LEAVE EXECADDRESS IN B
427 04C06 DD           BCEX   A              LEAVE RESCAN POINTER IN C
428 04C08 20           P=     0              LEAVE END-OF-BUFFER POINTER IN D
429 04C0A 04           SETHEX                RESET STATE
430 04C0C 03           RTNCC                 RETURN CARRY CLEAR
431                ****************************************************************
432                ****************************************************************
433                **
434                ** Name:(S) D=WORD  -  Read 8 Bytes And Convert To Uppercase
435                **
436                ** Category:   GENUTL
437                **
438                ** Purpose:
439                **      Read 8 bytes from memory and convert to uppercase.
440                **
441                ** Entry:
442                **      DO pointing at text to be read.
443                **
444                ** Exit:
445                **      P=0.
446                **      D[W] contains uppercase version of text.
447                **
448                ** Calls:      None.
449                **
450                ** Uses.......
451                **             C,D,P.
452                **
453                ** Stk lvls:   0
454                **
455                ** History:
456                **
457                **    Date     Programmer              Modification
458                **  --------   ----------   ------------------------------------
459                **             SA           Wrote
460                **  11/01/83   NM           Attempted to document
461                **
462                ****************************************************************
463                ****************************************************************
464 04C0E 1577 =D=WORD C=DAT1 W
465 04C12 AF7          D=C    W
466 04C15 20           P=     0
467 04C17 3F04         LCHEX  4040404040404040
           0404
           0404
           0404
           04
468 04C29 0E77         C=C&D  W
469 04C2D 81E          CSRB
470 04C30 0E77         C=C&D  W
471 04C34 BFE          C=-C-1 W
472 04C37 0E73         D=C&D  W
```

```
473 04C3B 01          RTN
474             ****************************************************************
475             ****************************************************************
476             **
477             ** Name:(S) SCAN    -  Scan LEXfile Text Table For Lexeme
478             **
479             ** Category:   PARUTL
480             **
481             ** Purpose:
482             **      Scan LEXfile text table for text matching keyword
483             **      machine is trying to parse.
484             **
485             ** Entry:
486             **      D[W] contains keyword machine is trying to parse (up
487             **        to 8 bytes).
488             **      D1 = input pointer (pointing at data which was read
489             **        into D[W].
490             **      D0 pointing at wordsize nibble of first keyword
491             **        to examine in text table.
492             **
493             ** Exit:
494             **      D1 moved past lexeme in input stream.
495             **      Carry set -> lexeme not found.
496             **      Carry clear -> token in A[A].
497             **
498             ** Calls:      None.
499             **
500             ** Uses.......
501             **              A[S],C,P,D0,D1.
502             **
503             ** Stk lvls:   0
504             **
505             ** History:
506             **
507             **    Date      Programmer              Modification
508             **  --------    ----------   ----------------------------------
509             **              SA           Wrote
510             **  11/01/83    NM           Added documentation
511             **
512             ****************************************************************
513             ****************************************************************
514 04C3D 161   SCAN10 D0=D0+ 2
515 04C40 1564 =SCAN   C=DAT0 S
516 04C44 80DF         P=C    15
517 04C48 160          D0=D0+ 1               MOVE TABLE POINTER TO TEXT
518 04C4B 1561         C=DAT0 WP              READ TEXT
519 04C4F 9E7          ?C>D   B               PAST WORD GROUP IN TABLE?
520 04C52 00           RTNYES                 IF SO, RETURN CARRY SET
521 04C54 136          CD0EX
522 04C57 809          C+P+1                  MOVE TABLE POINTER TO TOKEN
523 04C5A 136          CD0EX
524 04C5D 917          ?C#D   WP              LEXEME IDENTIFIED?
525 04C60 DD           GOYES  SCAN10          IF NOT, REPEAT SCAN10
526 04C62 137          CD1EX
527 04C65 809          C+P+1                  MOVE INPUT POINTER PAST LEXEME
```

```
528 04C68 137          CD1EX
529 04C6B D0           A=0    A
530 04C6D 14A          A=DATO B              READ TOKEN
531 04C70 03           RTNCC                 RETURN CARRY CLEAR
532             ****************************************************************
533 04C72 5    =FUNNY  NIBHEX 5              TEXT TABLE FOR "GO" WORDS
534 04C73 3555         NIBASC \SUB\
          24
535 04C79 00           CON(2) =tGOSUB
536 04C7B 3            NIBHEX 3
537 04C7C 45F4         NIBASC \TO\
538 04C80 00           CON(2) =tGOTO
539 04C82 1FF          NIBHEX 1FF
540             ****************************************************************
541 04C85 840   BLDVAR ST=0   0              CLEAR CHARACTERIZATION REQUEST
542 04C88 20           P=     0
543 04C8A 15B5         A=DAT1 6              READ 3 CHARACTERS
544 04C8E 31FD         LCHEX  DF             CONVERT FIRST TO UPPERCASE
545 04C92 0E66         A=A&C  B
546 04C96 3314         LCASC  \ZA\
          A5
547 04C9C 7000         GOSUB  =Range         LETTER?
548 04CA0 400          RTNC                  IF NOT, RETURN CARRY SET
549 04CA3 171          D1=D1+ 2
550 04CA6 AE8          B=A    B              SAVE FIRST IN B
551 04CA9 BF4          ASR    W              SHIFT OTHERS RIGHT
552 04CAC F4           ASR    A
553 04CAE 8F00         GOSBVL =DRANGE        DIGIT?
          000
554 04CB5 491          GOC    STRVAR         IF NOT, GOTO STRVAR
555 04CB8 171          D1=D1+ 2
556 04CBB BB1          BSL    X              SHIFT LETTER LEFT
557 04CBE F1           BSL    A
558 04CC0 3106         LCHEX  60
559 04CC4 AE5          B=C    B              CONSTRUCT ALPHADIGIT TOKEN
560 04CC7 A88          B=A    P
561 04CCA F4           ASR    A              SHIFT INPUT CHARACTERS DOWN
562 04CCC BB4          ASR    X
563 04CCF 7520  STRVAR GOSUB  STRCHK         STRING?
564 04CD3 401          GOC    ARGCHK         IF NOT, GOTO ARGCHK
565 04CD6 171          D1=D1+ 2
566 04CD9 F1           BSL    A              SHIFT TOKEN STREAM LEFT
567 04CDB BF1          BSL    W
568 04CDE 30D          LCHEX  D
569 04CE1 AE5          B=C    B              PLACE STRING TOKEN
570             ****************************************************************
571 04CE4 8E00  ARGCHK GOSUBL =GNXTCR        SKIP LEADING BLANKS
          00
572 04CEA 3182         LCASC  \(\
573 04CEE 966          ?A#C   B              PARENTHESIS?
574 04CF1 50           GOYES  ARGRTN         IF NOT, RETURN CARRY CLEAR
575 04CF3 852          ST=1   2              RECORD PARENTHESIS
576 04CF6 03    ARGRTN RTNCC                 RETURN CARRY CLEAR
577             ****************************************************************
578 04CF8 3142  STRCHK LCASC  \$\
```

```
579 04CFC 966          ?A#C   B            STRING?
580 04CFF 00           RTNYES              IF NOT, RETURN CARRY SET
581 04D01 851          ST=1   1            RECORD STRING
582 04D04 03           RTNCC               RETURN CARRY CLEAR
```

```
                        STITLE
                ********************************************************************
                ********************************************************************
                **
                ** Name:    STLXPT  -  Store LEXPTR
                **
                ** Category:   PARUTL
                **
                ** Purpose:
                **      Set LEXPTR to current value of D1.
                **
                ** Entry:
                **      None
                **
                ** Exit:
                **      (LEXPTR)  =  D1
                **      C(A)      =  D1
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: C(A)
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ---------------------------------
                **  11/02/82   SA & FH      Adapted from NTOKEN code
                **  12/14/82   SW           Added entry point for RESTAR
                **
                ********************************************************************
                ********************************************************************
04D06 137       =STLXPT CD1EX
04D09 1F00      =STLXP2 D1=(5) =LEXPTR
      000
04D10 145               DAT1=C A
04D13 135               D1=C
04D16 01                RTN

                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) NUMSCN  -  Scan Number In Lexical Analysis
                **
                ** Category:   PARUTL
                **
                ** Purpose:
                **      Generate token for numeric constant or solitary ASCII
                **      period.
                **
                ** Entry:
                **      D1 at start of numeric character string.
                **
```

```
637                ** Exit:
638                **      DEC mode.
639                **      P=0.
640                **      S3=1 for incomplete exponent.
641                **      D1 past numeric character string.
642                **      A[B] = numeric token and mantissa or ASCII digit.
643                **      B[M] = right-justified mantissa.
644                **      B[X] = exponent.
645                **
646                ** Calls:      DGTSTR, LDZERO, ROUND.
647                **
648                ** Uses.......
649                **             A,B,C,D,P,D1.
650                **
651                ** Stk lvls:   1
652                **
653                ** History:
654                **
655                **    Date      Programmer              Modification
656                **  --------    ----------   ----------------------------------
657                **  04/01/83    SA           Figured out register & subr usage
658                **  10/17/83    NM           Attempted to document
659                **
660                ********************************************************************
661                ********************************************************************
662 04D18 05    =NUMSCN SETDEC
663 04D1A 822          SB=0
664 04D1D AF0          A=0     W               INITIALIZE
665 04D20 AC3          D=0     S
666 04D23 2F           P=      15
667 04D25 304          LCHEX   4
668 04D28 AC5          B=C     S
669 04D2B 30D          LCHEX   D
670 04D2E 20           P=      0
671 04D30 7BE1         GOSUB   LDZERO          SKIP LEADING ZEROES
672 04D34 AB3          D=0     X               INITIALIZE NORMALIZER
673 04D37 A3F          D=D-1   X
674 04D3A B37   ILOOP  D=D+1   X               READ INTEGER MANTISSA
675 04D3D 7CE1         GOSUB   DGTSTR
676 04D41 58F          GONC    ILOOP
677 04D44 31E2         LCASC   \.\
678 04D48 966          ?A#C    B               CHECK FOR DECIMAL POINT
679 04D4B 34           GOYES   RNDCHK          IF NOT DECIMAL, GOTO RNDCHK
680 04D4D 171          D1=D1+  2
681 04D50 14B          A=DAT1  B
682 04D53 872          ?ST=1   2               PREVIOUS SIGNIFICANCE?
683 04D56 60           GOYES   FLOOP           IF SO, GOTO FLOOP
684 04D58 73C1         GOSUB   LDZERO          SKIP LEADING 0'S, ADJ NORMALIZER
685 04D5C 7DC1  FLOOP  GOSUB   DGTSTR          READ FRACTIONAL MANTISSA
686 04D60 5BF          GONC    FLOOP
687 04D63 871          ?ST=1   1               MANTISSA OK?
688 04D66 82           GOYES   RNDCHK          IF SO, GOTO RNDCHK
689 04D68 31E2         LCHEX   2E              SOLITARY PERIOD
690 04D6C AEA          A=C     B
691 04D6F 02           RTNSC                   RETURN
```

```
692
693 04D71 B46   RND     C=C+1  S          12 DIGITS
694 04D74 AE0          A=0    B
695 04D77 7E61         GOSUB  ROUND      ROUND MANTISSA
696 04D7B 14B          A=DAT1 B          REREAD NEXT CHARACTER
697 04D7E 5A1          GONC   EXPCHK     IF NO DECADE OFLO, GOTO EXPCHK
698 04D81 30C          LCHEX  C          RELOAD DIGIT COUNT
699 04D84 816          CSRC
700 04D87 B37          D=D+1  X          INCREMENT NORMALIZER
701 04D8A 6E00         GOTO   EXPCHK     GOTO EXPCHK
702 04D8E 94A  RNDCHK  ?C=0   S          NEED TO ROUND?
703 04D91 0E           GOYES  RND        IF SO, GOTO RND
704 04D93 BF0          ASL    W          ALIGN MANTISSA
705 04D96 BB4          ASR    X          REALIGN NEXT CHARACTER
706 04D99 AD8  EXPCHK  B=A    M          SAVE MANTISSA IN B(M)
707 04D9C ABB          C=D    X          SAVE NORMALIZER IN B(X)
708 04D9F AB5          B=C    X
709 04DA2 A3D          B=B-1  X          CORRECT NORMALIZER
710 04DA5 ACD          BCEX   S          PREPARE FOR POSSIBLE EXPONENT
711 04DA8 31FD         LCHEX  DF
712 04DAC 0E66         A=A&C  B
713 04DB0 3154         LCASC  \E\
714 04DB4 966          ?A#C   B          EXPLICIT EXPONENT?
715 04DB7 31           GOYES  BLDLNK     IF NOT, GOTO BLDLNK
716 04DB9 137          CD1EX             SAVE INPUT POINTER IN D
717 04DBC D7           D=C    A
718 04DBE 137          CD1EX
719 04DC1 6210         GOTO   SLOOP      GOTO SLOOP
720 04DC5 DB   EXPERR  C=D    A          RESTORE INPUT POINTER TO "E"
721 04DC7 135          D1=C
722 04DCA 6880 BLDLNK  GOTO   BUILD      GOTO BUILD
723 04DCE BCF  SLOOP-  D=-D-1 S          COMPLEMENT EXPONENT SIGN
724 04DD1 853  SLOOP+  ST=1   3          RECORD EXISTENCE OF EXPONENT SIGN
725 04DD4 171  SLOOP   D1=D1+ 2          READ NEXT CHARACTER
726 04DD7 14B          A=DAT1 B
727 04DDA 31D2         LCASC  \-\
728 04DDE 962          ?A=C   B          MINUS SIGN?
729 04DE1 DE           GOYES  SLOOP-     IF SO, REPEAT SLOOP-
730 04DE3 30B          LCHEX  B
731 04DE6 962          ?A=C   B          PLUS SIGN?
732 04DE9 8E           GOYES  SLOOP+     IF SO, REPEAT SLOOP+
733 04DEB 841          ST=0   1          PREPARE FOR READING EXPONENT
734 04DEE 842          ST=0   2
735 04DF1 D0           A=0    A
736 04DF3 7821         GOSUB  LDZERO     SKIP LEADING ZEROES
737 04DF7 7231 ELOOP   GOSUB  DGTSTR     READ EXPONENT
738 04DFB 5BF          GONC   ELOOP
739 04DFE 861          ?ST=0  1          EXPONENT OK?
740 04E01 4C           GOYES  EXPERR     IF NOT, GOTO EXPERR
741 04E03 843          ST=0   3          EXPONENT SIGN OK
742 04E06 862          ?ST=0  2          ZERO EXPONENT?
743 04E09 A4           GOYES  BUILD      IF SO, GOTO BUILD
744 04E0B F4           ASR    A
745 04E0D F4           ASR    A
746 04E0F 3200         LCHEX  500
```

```
          5
747 04E14 94F       ?D#0   S          NEGATIVE EXPONENT?
748 04E17 22        GOYES  MINUSX      IF SO, GOTO MINUSX
749 04E19 94A PLUSX ?C=0   S          TOO MANY DIGITS IN EXPONENT?
750 04E1C 41        GOYES  BIG        IF SO, GOTO BIG
751 04E1E 9A6       ?A>C   XS         STATED EXPONENT TOO BIG?
752 04E21 F0        GOYES  BIG        IF SO, GOTO BIG
753 04E23 A38       B=A+B  X          ADD EXPONENT TO NORMALIZER
754 04E26 9B5       ?B<C   X          TRUE EXPONENT < 500?
755 04E29 A2        GOYES  BUILD      IF SO, GOTO BUILD
756 04E2B 928       ?A=0   XS         STATED EXPONENT < 100?
757 04E2E 52        GOYES  BUILD      IF SO, GOTO BUILD
758 04E30 3100 BIG  LC(2)  =tBIG
759 04E34 AEA       A=C    B          "TOO BIG" TOKEN
760 04E37 03        RTNCC             RETURN CARRY CLEAR
761 04E39 94A MINUSX ?C=0  S          TOO MANY DIGITS IN EXPONENT?
762 04E3C 96        GOYES  DNORM3     IF SO, GOTO DNORM3
763 04E3E 9A6       ?A>C   XS         STATED EXPONENT TOO BIG?
764 04E41 46        GOYES  DNORM3     IF SO, GOTO DNORM3
765 04E43 B38       B=B-A  X          SUBTRACT EXPONENT FROM NORMALIZER
766 04E46 9B1       ?B>C   X          TRUE EXPONENT > -500?
767 04E49 A0        GOYES  BUILD      IF SO, GOTO BUILD
768 04E4B B39       C=C-B  X
769 04E4E 92D       ?B#0   XS         TRUE EXPONENT < 100?
770 04E51 34        GOYES  DNORM2     IF SO, GOTO DNORM2
771 04E53 AB0 BUILD A=0    X
772 04E56 959       ?B=0   M          ZERO MANTISSA?
773 04E59 A2        GOYES  ZERO       IF SO, GOTO ZERO
774 04E5B AC4       A=B    S
775 04E5E 810       ASLC
776 04E61 3221      LCHEX  012
          0
777 04E66 B32       C=C-A  X
778 04E69 04        SETHEX
779 04E6B B04       A=A+1  P
780 04E6E BE0       ASL    B
781 04E71 AD4       A=B    M
782 04E74 935       ?B#C   X          FLOATING-POINT CONSTANT?
783 04E77 21        GOYES  FLOAT      IF SO, GOTO FLOAT
784 04E79 93D       ?B#0   X          MULTI-DIGIT INTEGER?
785 04E7C 01        GOYES  NDIGIT     IF SO, GOTO NDIGIT
786 04E7E F4        ASR    A
787 04E80 BB4       ASR    X          CONSTRUCT ASCII DIGIT
788 04E83 B24 ZERO  A=A+1  XS
789 04E86 B24       A=A+1  XS
790 04E89 B24 FLOAT A=A+1  XS
791 04E8C BF4 NDIGIT ASR   W
792 04E8F 03        RTNCC             RETURN CARRY CLEAR
793
794 04E91 BD1 DNORM1 BSL   M          NORMALIZE
795 04E94 A4D DNORM2 B=B-1 S
796 04E97 94D       ?B#0   S
797 04E9A 7F        GOYES  DNORM1
798 04E9C 822       SB=0
799 04E9F AF0       A=0    W
```

```
800 04EA2 AD4          A=B    M
801 04EA5 BF4   DNORM3 ASR    W              DENORMALIZE
802 04EA8 A3E          C=C-1  X
803 04EAB 59F          GONC   DNORM3
804 04EAE 7730         GOSUB  ROUND          ROUND
805 04EB2 AD8          B=A    M
806 04EB5 3210         LCHEX  501
          5
807 04EBA AB5          B=C    X              SET EXPONENT
808 04EBD 321D         LCHEX  1D1            LOAD 1E-499 TOKEN
          1
809 04EC2 412          GOC    DNORM5         IF ROUNDING OVERFLOW, GOTO DNORM5
810 04EC5 3211         LCHEX  D11
          D
811 04ECA 958          ?A=0   M              ZERO MANTISSA?
812 04ECD 71           GOYES  DNORM5         IF SO, GOTO DNORM5
813 04ECF 3211         LCHEX  111            LOAD SMALL TKN & INITIALIZE CNT
          1
814 04ED4 04           SETHEX
815 04ED6 BF4   DNORM4 ASR    W              SHORTEN
816 04ED9 B26          C=C+1  XS
817 04EDC 928          ?A=0   XS
818 04EDF 7F           GOYES  DNORM4
819 04EE1 BF0          ASL    W
820 04EE4 ABA   DNORM5 A=C    X              SET TOKEN
821 04EE7 03           RTNCC                 RETURN CARRY CLEAR
822                    ********************************************************************
823 04EE9 A34   ROUND  A=A+A  X              OBVIOUSLY ROUND DOWN?
824 04EEC 500          RTNNC                 IF SO, RETURN CARRY SET
825 04EEF 93C          ?A#0   X              OBVIOUSLY ROUND UP?
826 04EF2 70           GOYES  RND10          IF SO, GOTO RND10
827 04EF4 832          ?SB=0                 STUFF OFF THE END?
828 04EF7 D0           GOYES  RND20          IF SO, GOTO RND20
829 04EF9 B54   RND10  A=A+1  M              INCREMENT MANTISSA
830 04EFC 500          RTNNC                 IF DECADE OFLO, RETURN CRY CLEAR
831 04EFF B54          A=A+1  M              CREATE UNIT MANTISSA
832 04F02 02           RTNSC                 RETURN CARRY SET
833 04F04 3400  RND20  LCHEX  01000          LOAD DIAGNOSTIC CONSTANT
          010
834 04F0B 0EF2         C=A&C  A              ISOLATE BOTTOM BIT OF MANTISSA
835 04F0F 8AE          ?C#0   A              BOTTOM BIT ON?
836 04F12 7E           GOYES  RND10          IF SO, GOTO RND10
837 04F14 03           RTNCC                 RETURN CARRY CLEAR
838                    ********************************************************************
839 04F16 851   RPTLDZ ST=1   1              RECORD EXISTENCE OF DIGIT
840 04F19 A3F          D=D-1  X              DECREMENT NORMALIZER
841 04F1C 171          D1=D1+ 2              INCREMENT TEXT POINTER
842 04F1F 3103  LDZERO LCASC  \0\
843 04F23 14B          A=DAT1 B              READ CHARACTER
844 04F26 962          ?A=C   B              ZERO?
845 04F29 DE           GOYES  RPTLDZ         IF SO, REPEAT RPTLDZ
846 04F2B 01           RTN                   RETURN
847                    ********************************************************************
848 04F2D 3193  DGTSTR LCASC  \9\
849 04F31 9E6          ?A>C   B              IS CHARACTER A DIGIT?
```

```
850 04F34 00          RTNYES                IF NOT, RETURN WITH CARRY SET
851 04F36 3103        LCASC  \0\
852 04F3A 9E2         ?A<C   B
853 04F3D 00          RTNYES
854 04F3F 94A         ?C=0   S              SIGNIFICANCE EXCEEDED?
855 04F42 11          GOYES  NXTDGT         IF SO, GOTO NXTDGT
856 04F44 851         ST=1   1              RECORD EXISTENCE OF DIGIT
857 04F47 852         ST=1   2              RECORD SIGNIFICANCE
858 04F4A A4E         C=C-1  S              ADJUST DIGIT COUNT
859 04F4D BE0         ASL    B              SHIFT DIGIT INTO MANTISSA
860 04F50 BF0         ASL    W
861 04F53 B84   NXTDGT ASR    P              SET STICKY BIT IF APPROPRIATE
862 04F56 171         D1=D1+ 2              INCREMENT TEXT POINTER
863 04F59 14B         A=DAT1 B              READ NEXT CHARACTER
864 04F5C 03          RTNCC                 RETURN CARRY CLEAR
865 04F5E             END
```

```
=ALLDUN  Abs   19439 #04BEF -   420   401   409   413
 ARGCHK  Abs   19684 #04CE4 -   571   395   564
 ARGRTN  Abs   19702 #04CF6 -   576   574
 ARRAY   Ext               -   414
 AWFUL   Abs   18789 #04965 -   210   220
 BFRERR  Abs   18978 #04A22 -   266   271   273
 BIG     Abs   20016 #04E30 -   758   750   752
 BLDLNK  Abs   19914 #04DCA -   722   715
 BLDVAR  Abs   19589 #04C85 -   541   380   407
 BUILD   Abs   20051 #04E53 -   771   722   743   755   757   767
 CHARLY  Abs   19444 #04BF4 -   422   378
 CORUPT  Ext               -   266
=D=WORD  Abs   19470 #04C0E -   464   283   370
 DGTSTR  Abs   20269 #04F2D -   848   181   675   685   737
 DNORM1  Abs   20113 #04E91 -   794   797
 DNORM2  Abs   20116 #04E94 -   795   770
 DNORM3  Abs   20133 #04EA5 -   801   762   764   803
 DNORM4  Abs   20182 #04ED6 -   815   818
 DNORM5  Abs   20196 #04EE4 -   820   809   812
 DRANGE  Ext               -   553
 ELOOP   Abs   19959 #04DF7 -   737   738
 EXIT1   Abs   18913 #049E1 -   248   211   237
 EXIT2   Abs   19363 #04BA3 -   396   390
 EXPCHK  Abs   19865 #04D99 -   706   697   701
 EXPERR  Abs   19909 #04DC5 -   720   740
 FCN     Abs   19339 #04B8B -   388   347
 FLOAT   Abs   20105 #04E89 -   790   783
 FLOOP   Abs   19804 #04D5C -   685   683   686
 FN      Ext               -   383
=FN-GO   Abs   19249 #04B31 -   359
 FNERR   Abs   19393 #04BC1 -   406   381
 FNSCAN  Abs   19312 #04B70 -   380   366
 FUNCDO  Ext               -   278   422
=FUNNY   Abs   19570 #04C72 -   533   371
 GARRR   Abs   19146 #04ACA -   321   358
 GARRRj  Abs   19246 #04B2E -   358   374
 GNXTCR  Ext               -   171   197   367   571
=HOWARD  Abs   19442 #04BF2 -   421
 ILOOP   Abs   19770 #04D3A -   674   676
 IOFNDO  Ext               -   270
 L#0000  Abs   18735 #0492F -   192   186
 LDZERO  Abs   20255 #04F1F -   842   175   671   684   736
=LEAVE   Abs   19457 #04C01 -   425
 LEX     Abs   19088 #04A90 -   300   291
 LEXPTR  Ext               -   618
 LLOOP   Abs   18702 #0490E -   180   182
 LXTYPT  Ext               -   224
 MINUSX  Abs   20025 #04E39 -   761   748
 MISC    Abs   18861 #049AD -   229
 NDIGIT  Abs   20108 #04E8C -   791   785
 NOT-CR  Abs   18796 #0496C -   212   208
=NTOKEN  Abs   18747 #0493B -   197   184
=NTOKNL  Abs   18662 #048E6 -   166
 NUMBER  Abs   18875 #049BB -   234   230
=NUMSCN  Abs   19736 #04D18 -   662   242
```

```
 NXTDGT  Abs   20307 #04F53 -   861   855
 NXTROM  Abs   19070 #04A7E -   293   319
 OKCHAR  Abs   18833 #04991 -   221   214
 PLUSX   Abs   19993 #04E19 -   749
=PRESCN  Abs   19017 #04A49 -   277
 RELOP   Abs   18946 #04A02 -   257   239
 RELOP1  Abs   18918 #049E6 -   250   262
 RELOP2  Abs   18936 #049F8 -   254   259
=RESCAN  Abs   19020 #04A4C -   278
 RESTRT  Abs   18737 #04931 -   193   188
 RND     Abs   19825 #04D71 -   693   703
 RND10   Abs   20217 #04EF9 -   829   826   836
 RND20   Abs   20228 #04F04 -   833   828
 RNDCHK  Abs   19854 #04D8E -   702   679   688
 ROMNUM  Abs   19370 #04BAA -   398   355
 ROUND   Abs   20201 #04EE9 -   823   695   804
 RPTLDZ  Abs   20246 #04F16 -   839   845
 Range   Ext               -   213   218   261   547
=SCAN    Abs   19520 #04C40 -   515   318   372
 SCAN10  Abs   19517 #04C3D -   514   525
 SCNADR  Abs   19132 #04ABC -   316   310
 SETUP   Abs   19232 #04B20 -   353   349
=SHFTKN  Abs   19430 #04BE6 -   417   386   405
 SLOOP   Abs   19924 #04DD4 -   725   719
 SLOOP+  Abs   19921 #04DD1 -   724   732
 SLOOP-  Abs   19918 #04DCE -   723   729
=STLXP2  Abs   19721 #04D09 -   618
=STLXPT  Abs   19718 #04D06 -   617   172   204
 STRCHK  Abs   19704 #04CF8 -   578   394   563
 STRVAR  Abs   19663 #04CCF -   563   554
 VBLLNK  Abs   19066 #04A7A -   292   298
=VRIABL  Abs   19396 #04BC4 -   407   292
 WEIRD   Abs   19242 #04B2A -   356   343
 WORD    Abs   18984 #04A28 -   267   235
 ZERO    Abs   20099 #04E83 -   788   773
 bLEX    Ext               -   269
 oSPDTB  Ext               -   289
 oSPDn2  Ext               -   305
 tARRAY  Ext               -   416
 tBIG    Ext               -   758
 tEOL    Ext               -   209
 tFN     Ext               -   385
 tGOSUB  Ext               -   535
 tGOTO   Ext               -   538
 tRELOP  Ext               -   263
 tXFN    Ext               -   397
 tXWORD  Ext               -   354
```

Input Parameters

  Source file name is AB&LEX::MS

  Listing file name is AB/LEX:TI:ML::-1

  Object file name is AB%LEX:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS    GGG    &     L      DDDD    CCC
 2            *    S   S  G   G  & &    L       D  D  C   C
 3            *    S      G      & &    L       D  D  C
 4            *     SSS   G GGG   &     L       D  D  C
 5            *        S  G   G  & & &  L       D  D  C
 6            *    S   S  G   G  &  &   L       D  D  C   C
 7            *     SSS    GGG    && &  LLLLL  DDDD    CCC
 8                      TITLE  Line Decompile <831216.1615>
 9 04F5E                ABS    #04F5E
10                      RDSYMB TIXEQU::MS
11           *
12           ***************************************************************
13           ***************************************************************
14           **
15           ** Name:(S) LDCOMP  -  Line Decompile Driver
16           ** Name:(S) LDCM10  -  Line Decompile Driver
17           ** Name:(S) LDCEXT  -  Line Decompile Driver
18           ** Name:(S) LDSST1  -  Line Decompile Driver
19           ** Name:(S) LDSST2  -  Line Decompile Driver
20           **
21           ** Category:   DCMUTL
22           **
23           ** Purpose:
24           **      LINE DECOMPILE DRIVER
25           **
26           ** Entry:
27           **      P=0
28           **      D1 @ BEGINNING OF COMPILED LINE IN RAM.
29           **
30           **      LDCOMP:  1) Updates current line
31           **               2) Clears SST flag
32           **               3) Decompiles entire BASIC line
33           **      LDCM10:  Does 2 & 3 above
34           **      LDCEXT:  Same as LDCM10, EXCEPT that this is
35           **               used to 'externally invoke' decompile.
36           **               Any memerr will return control to caller
37           **               with carry set.
38           **      sSSTdc=1 => only decompiles 1 stmt at a time
39           **      LDSST1:  SST entry for Decompile w/ Line#
40           **               Assumes sSSTdc (S1) set appropriately
41           **      LDSST2 : SST entry for Multi-stmt Line
42           **               Assumes sSSTdc (S1) set appropriately
43           **
44           ** Exit:
45           **      Normal entry:
46           **      Carry Clear   (through LSTLEN exit)
47           **      Decompiled Line sent to Input/Output Buffer
48           **      R0 past tEOL
49           **      If LDSST1/LDSST2 entry
50           **         D1 @ Tokenized Statement Terminator
51           **      B(A) = BUFFER LENGTH (#CHARACTERS)
52           **      Output Buffer collasped --> AVMEMS <-- OUTBS
53           **
54           **      If LDCEXT entry is used:
55           **      Carry clear => normal exit
```

```
                **                      DO past ascii stream
                **                      OUTBS is start of ascii stream
                **                      A(A) past tEOL of line decompiled
                **      Carry set   => Memerr
                **
                ** Calls:      RTNSET, SAVELN, LDCSET, LIN#DC, GTXT+1, AD1+2,
                **             'DC (OUTBYT), ASCICK, !TEST, GTEXTI, OUTNBC
                **             LIN#AU
                **
                ** Uses.......
                **  Exclusive: A-D, D1,D0, S0,S3,S5,S6,S7,S8,sSSTdc (S1)
                **  Inclusive: A-D, D1,D0, S0,S3,S5,S6,S7,S8,S1, R0-R2,
                **             S-R0-2 & flRTN  (if LDCEXT entry used)
                **
                **      sSSTdc = SST Decompile - GLOBAL throughout decompile
                **      S6 (VARDC),S8,CURRL
                **
                **      R0 = Pointer past tEOL (provided LDCEXT not used)
                **      R1 = Preserved D1
                **      R2 = Main Table Address
                **      R3 cannot be used, it is used by "LIST"
                **
                **      SAVES NEW LINE# INTO CURRL  (UPDATES CURRENT LINE)
                **      CLEARS S8 FOR ALL BEGIN BASIC DECOMPILE STATEMENTS:
                **      (CURRENTLY USED BY LISTDC & USER/BEEPDC)
                **
                ** Stk lvls:   6
                **
                ** Note:
                **
                **      No single Decompile routine can used more then 6 lvls
                **      EXPRDC uses 4 subroutine levels
                **
                **      sSSTdc (S1) must not be used by individual Decompile
                **        routines
                **
                **      Any decompile routine that POLLs must set AVMEMS at
                **      the Current DO (call AVS=DO).  This prevents the Poll
                **      Save area from overwriting the Output Buffer.
                **      Decompile, on exit, will set AVMEMS back @ OUTBS.
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ---------------------------------
                **  07/13/82   J.P.         Modified documentation
                **  08/30/82   J.P.         Fixed SST/ELSE decompile
                **
                ********************************************************************
                ********************************************************************
                *
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) RELJMP  -  Relative Jump From (D1)
```

```
111             **
112             ** Category:   GENUTL
113             **
114             ** Purpose:
115             **      RELJMP reads the address pointed to by D1, adds it to
116             **      D1, then does a direct jump to the resulting address.
117             **
118             **      The mainframe uses RELJMP to jump to a decompile
119             **      routine.
120             **
121             ** Entry:
122             **      D1 points to relative address
123             **
124             ** Exit:
125             **      D1 = R1 on entry
126             **      A(5-0) = 6 nibbles pointed to by D1
127             **      C=A
128             **      PC is at resulting address
129             **
130             ** Calls:      none
131             **
132             ** Uses:       A,C,D1
133             **
134             ** Stk lvls:   0
135             **
136             ** Detail:
137             **      When the mainframe uses RELJMP to decompile a
138             **      statement, on entry D1 points to the decompile address
139             **      and R1 contains the pointer into the token stream, ie
140             **      R1 points past the begin BASIC token.  So on exit from
141             **      RELJMP (upon entry to the decompile routine), D1 points
142             **      past the begin BASIC token and A contains the first six
143             **      tokenized nibbles that follow.
144             **
145             ** History:
146             **
147             **    Date     Programmer             Modification
148             **  --------   ----------   ----------------------------------
149             **  11/08/83   S.W.         Added documentation header
150             **
151             ********************************************************************
152             ********************************************************************
153             *
154 04F5E 07    =LDCEXT C=RSTK                  Pull rtn address off stack
155 04F60 8E00          GOSUBL =RTNSET          Set system flag flRTN
          00
156 04F66 580           GONC   LDCM10           (B.E.T.) for hard return
157             * Clear Single Step Decompile flag
158             * Save Line#
159             * Set D = End of Available Memory
160             * Set D0 = Output Buffer for Decompile = End of Program Memory
161             *
162 04F69 8E00  =LDCOMP GOSUBL =savel#          SAVE LINE#
          00
163 04F6F 841   =LDCM10 ST=0   sSSTdc           Clear SST decompile flag
```

```
164                *
165                * SST Entry for Decompile @ Line#
166                *
167 04F72 7AE0 =LDSST1 GOSUB  LDCSET         Set D, DO
168                *
169                * Decompile Line#
170                * Save position @ space after Line# for Cursor position (LDCSPC)
171                * Save Line length byte in PCADDR
172                *
173 04F76 7B91         GOSUB  LIN#DC
174 04F7A 136          CDOEX                Save position after Line#
175 04F7D 1B00         DO=(5) =LDCSPC
           000
176 04F84 144          DAT0=C A
177 04F87 133   LDCIF  AD1EX
178 04F8A 1B00         DO=(5) =INADDR
           000
179 04F91 140          DAT0=A A             WRITE ADDR OF LINE LENGTH BYTE
180 04F94 131          D1=A                 RESTORE D1
181 04F97 134          DO=C                 Restore DO
182 04F9A 7322         GOSUB  GTXT+1        OUTPUT BLANK AFTER LINE#
183                *
184                * SST Entry for Decompile at Multi-statement Line
185                *
186 04F9E 78D6 =LDSST2 GOSUB  AD1+2         Step over stmt length (or tEXTIF)
187 04FA2 848          ST=0   8             CLEAR FOR BASIC DECOMPILE ROUTINES
188 04FA5 849          ST=0   9
189                *
190                * If Label
191                *    Label statements are tokenized as:
192                *      tLBLST <5 nib Label chain> <ASCII Label name> (t@ | tEOL)
193                *    Labels are decompiled
194                *      " (ASCII Label name) ":
195                *    Decompile label statement
196                *    Output Closing quote and ":"
197                *    If EOL follows  (Isolated Label statement)
198                *       go Terminate Output buffer and return
199                *    else
200                *       Save Address of Statement Length byte
201                *
202 04FA8 3100         LC(2)  =tLBLST       Label Statement Token
203 04FAC 966          ?A#C   B             Not Label Statement ?
204 04FAF F2           GOYES  LDCM30
205 04FB1 176          D1=D1+ 7             Skip Label Token & label chain
206 04FB4 7C27         GOSUB  'DC           Output quote before label
207 04FB8 7291         GOSUB  ASCICK        Read label and output
208 04FBC 7427         GOSUB  'DC
209 04FC0 7827         GOSUB  :DC           Output closing quote & ":"
210 04FC4 14B          A=DAT1 B             Check for End of Label statement
211 04FC7 3100         LC(2)  =tEOL
212 04FCB 966          ?A#C   B             Not at end ?
213 04FCE 60           GOYES  LDCM20
214 04FD0 6D83         GOTO   OUTEOL        Terminate buffer and return
215                *
216                * Multi-statement entry
```

```
                *   DO positioned at @
                *   Statement byte follows
                *
04FD4 136    LDCM20 CDOEX                SAVE OUTPUT PTR IN C
04FD7 171           D1=D1+ 2             STEP OFF '@'
04FDA 6CAF          GOTO   LDCIF
                *
                * If Variable
                *    go Decompile Implied LET Statement
                *
04FDE D8     LDCM30 B=A    A
04FE0 3100          LC(2)  =LASTFN        Limit of MAINT
04FE4 9E6           ?A>C   B              Not an implied LET ?
04FE7 60            GOYES  LDCM50
04FE9 6482          GOTO   LETDC          LET Decompile
                *
                * Check for "!" on start of line
                *
04FED 7524   LDCM50 GOSUB  !TEST
04FF1 560           GONC   LDCM60
04FF4 6053          GOTO   !DC
                *
                * Skip token and Get Text for token
                *
04FF8 7A70   LDCM60 GOSUB  GTEXTI         Get Text for token
04FFC 5B3           GONC   LDCM80         Main keyword | XWORD Text found
                *
                * LEX ID not found
                *    Output "XWORD"
                *    Convert and Output LEX ID, Entry# to Decimal ASCII
                *    Order in RAM: LEX ID, Entry#
                *    Entry to subroutine:
                *       R0(3-2) = LEX ID, R0(B) = Entry#
                *    Skip rest of statement and Return
                *
04FFF 3985          LCASC  \DROWX\        Output "XWORD"
      75F4
      2544
0500B 29            P=     9
0500D 7214          GOSUB  OUTNBC
05011 14B           A=DAT1 B              Read LEX ID
05014 F0            ASL    A              Shift to A(2-3)
05016 F0            ASL    A
05018 7E56          GOSUB  AD1+2          Read in entry#
0501C 100           R0=A                  Must be in R0 for subroutine
0501F F4            ASR    A
05021 F4            ASR    A
05023 2E            P=     14             Convert HEX to DEC
05025 79F0          GOSUB  LIN#AU         Output 2 digits, supp. lding 0's
05029 110           A=R0
0502C 2C            P=     12             HEX => DEC - output 2 digits
0502E 70F0          GOSUB  LIN#AU         suppress up to 4 leading zeroes.
05032 8C2C          GOLONG SKIPDC         Skip rest of statement
      70
                *
```

```
269            * Output Keyword Text & Blank
270            *
271 05038 80DF  LDCM80 P=C    15            (#nibbles-1) of Text
272 0503C 76E3         GOSUB  OUTNBS
273 05040 7D71         GOSUB  GTXT+1
274            *
275            * D1 @ Execution Address
276            * Compute Decompile Address, Push on Stack
277            *
278 05044 1C9          D1=D1- 10             Move to Decompile Adress
279 05047 147  =RELJMP C=DAT1 A              Read Rel. Decompile Address
280 0504A 133          AD1EX                 Position in Table --> A
281 0504D C2           C=C+A  A              Absolute Decompile Address
282 0504F 06           RSTK=C                Push decompile address on H/W stack
283            *
284            * Restore D1 to Input Line
285            * Read next token
286            * Jump to Decompile Routine
287            *
288 05051 111          A=R1
289 05054 131          D1=A                  Restore ptr to Input Line
290 05057 15B5         A=DAT1 6              Read next token
291 0505B AF6          C=A    W
292 0505E 01           RTN                   JUMP TO DECOMPILE ROUTINE
```

```
                        STITLE LDCSET - Set Decompile bounds
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:(S) LDCSET  -  Set D=AVMEME; DO=OUTBS
                    ** Name:    DO=OBS  -  Set D=AVMEME; DO=OUTBS
                    **
                    ** Category:   PTRUTL
                    **
                    ** Purpose:
                    **      Set D @ AVMEME, DO @ OUTBS
                    **
                    ** Entry:
                    **      2 entry points:
                    **              1) LDCSET - Sets D(A) to AVMEME. Sets DO to OUTBS.
                    **              2) DO=OBS - Sets DO to OUTBS.
                    **
                    ** Exit:
                    **    All entry points:
                    **      C(A)   =  (OUTBS)
                    **      DO     @  (OUTBS)
                    **      Carry  =  Entry state
                    **    LDCSET only:
                    **      D(A)   =  (AVMEME)
                    **
                    ** Calls:      D=AVME
                    **
                    ** Uses.......
                    **  Exclusive: C(A), DO, D(A) (LDCSET only)
                    **
                    ** Stk lvls:   1 (LDCSET), 0 (DO=OBS and DO=OUTB)
                    **
                    ** Detail:
                    **      The carry must be PRESERVED due to call from AUTO
                    **
                    ** History:
                    **
                    **    Date     Programmer             Modification
                    **  --------   ----------   ----------------------------------
                    **  07/13/82   J.P.         Modified documentation
                    **
                    ****************************************************************
                    ****************************************************************
05060 8F00 =LDCSET GOSBVL =D=AVME        Set D(A) to AVMEME
      000
05067      =DO=OBS
05067 1B00 =DOOUTB DO=(5) =OUTBS           GET INPUT BUF PTR
      000
0506E 146          C=DAT0 A
05071 134          DO=C
05074 01           RTN                   The Carry must be UNTOUCHED
```

```
342                 STITLE Get Text for Keyword
343               ********************************************************************
344               ********************************************************************
345               **
346               ** Name:(S) GTEXT   -  Get Text for Keyword/Function
347               ** Name:    GTEXTM  -  Get Text for Keyword/Function
348               ** Name:    GTEXTX  -  Get Text for Keyword/Function
349               **
350               ** Category:   DCMUTL
351               **
352               ** Purpose:
353               **      Get Text for Keyword or Function
354               **
355               ** Entry:
356               **      D0 pointing into output buffer
357               **      D(A) contains available memory end (AVMEME)
358               **      GTEXTI:  A = Main token | XWORD token | XFN token
359               **               D1 = At Keyword | Function token
360               **               D1 incremented by 2 on entry
361               **               P=0
362               **
363               **      GTEXT:   A = Main token | XWORD token | XFN token
364               **               D1 = Past Keyword | Function token
365               **               P=0
366               **
367               **      GTEXTM:  Mainframe Lex Table used
368               **               D1 = Past Keyword | Function token
369               **               P=0
370               **
371               **      GTEXTX:  XWORD Lex Tables used
372               **               D1 = Past XWORD | XFN token
373               **
374               ** Exit:
375               **      P=0
376               **      Carry Clear
377               **           A = Text
378               **         C(S)= # nibbles - 1
379               **          D1 @ Execution address for token
380               **          R1 = D1 on entry            (Past token)
381               **               D1+4 on entry if XWORD  (Past Lex ID and Entry#)
382               **
383               **      Carry Set
384               **          XWORD | XFN not found
385               **          D1 @ D1 on entry            ( @ Lex ID )
386               **
387               ** Calls:      XMTADR, MTADR+
388               **
389               ** Uses.......
390               **  Exclusive: A,B,C,R1,R2,D1
391               **  Inclusive: A,B,C,R1,R2,D1
392               **
393               **      R1 = D1 @ entry
394               **      R2 = Main Table Address
395               **
396               ** Stk lvls:   2
```

```
397             **
398             ** Algorithm:
399             **
400             ** GTEXTI: Increment D1 past token
401             ** GTEXT:  If XWORD or XFN
402             **             goto GTEXTX
403             ** GTEXTM: Load MAINT address
404             **         Save token --> B(A)
405             **         goto 1;
406             ** GTEXTX: Read LEX ID, Entry#
407             **         Calculate Main Table address (XMTADR)
408             **         If address NOT found ---> RTNC
409             **         Skip over LEX ID and Entry#
410             **       1: Save D1       (R1)
411             **         Save Main Table Address  (R2)
412             **         Calculate Start of Text Table
413             **           Txt Tbl Rel Addr Ptr @ Main Table Addr - oSPDn2 + 1
414             **           Txt Tbl Start = Txt Tbl Ptr + (Txt Tbl Ptr)
415             **         D1 <-- Text Table Start
416             **         C  <-- Main Table Address  (R2)
417             **         Calculate Entry into Main Table  (MTADR+)
418             **         Read Text Table Offset
419             **         Read Execution Address & Save it (R2)
420             **         Compute Entry into Text Table
421             **         Read # nibbles for text          (C(S))
422             **         Read ASCII Text                  (A)
423             **         Set D1 = Execution address       (R2)
424             **         RTNCC
425             **
426             ** History:
427             **
428             **    Date      Programmer              Modification
429             **  --------   ----------   ----------------------------------
430             **  07/13/82   J.P.         Modified documentation
431             **  08/17/82   S.W.         Added GTEXTI entry point
432             **  12/06/82   J.P.         Fixed XWORD not found exit conditions
433             **
434             ********************************************************************
435             ********************************************************************
436 05076 171   GTEXTI D1=D1+ 2
437 05079 3100 =GTEXT  LC(2)  =tXWORD
438 0507D 962         ?A=C   B
439 05080 B1          GOYES  GTEXTX
440 05082 3100        LC(2)  =tXFN
441 05086 962         ?A=C   B
442 05089 21          GOYES  GTEXTX
443 0508B 3400 =GTEXTM LC(5)  =MAINT         Mainframe Main Table address
          000
444 05092 D1          B=0    A
445 05094 AE8         B=A    B               Move token to B
446 05097 6210        GOTO   GTXT10
447             *
448             * XWORD | XFN entry
449             *    Read LEX ID, Entry#
450             *    Extract Main Table Address  (XMTADR)
```

```
451             *    If address NOT found --->   RTNC
452             *    Skip LEX ID and Entry#
453             *
454 0509B 143   =GTEXTX A=DAT1 A              Read ROM ID, Entry#
455 0509E 8E00          GOSUBL =XMTADR        Calc Main Table Address
          00
456 050A4 400           RTNC                  Address not found
457 050A7 173           D1=D1+ 4              Skip ROM ID, Entry#
458             *
459             * Calculate Start of Text Table
460             *
461             * Text Table Rel Addr Ptr = Main Table addr - oSPDn1 + 1
462             * Text Table Start = Text Table Ptr + (Text Table Ptr)
463             *
464             *
465 050AA 133    GTXT10 AD1EX                 Save D1
466 050AD 101           R1=A
467 050B0 10A           R2=C                  Save Main Table Address
468 050B3 DA            A=C    A
469 050B5 D2            C=0    A
470 050B7 30E           LC(1)  =oSPDn2        Offset to Speed Nibble 2
471 050BA EA            A=A-C  A              One nibble above Text Table
472 050BC 131           D1=A
473 050BF 147           C=DAT1 A              Read Speed nibble, Text Table ptr
474 050C2 F6            CSR    A              Shift off speed nibble
475 050C4 CA            A=A+C  A              Abs. Text Table start
476 050C6 131           D1=A                  Save it
477             *
478             * Calculate Entry Address into Main Table
479             *     C=Address,  B=Offset
480             * Exit from MTADR+
481             *     C = Old D1, D1=Entry into Main Table
482             *
483 050C9 11A           C=R2                  Main Table address
484 050CC 8E00          GOSUBL =MTADR+        Calc Entry address
          00
485 050D2 D5            B=C    A              Text Table Start (old D1)
486             *
487             * Read Text Table offset within Main Table
488             * Read Rel Execution Address, Calc ABS addr, Save it  (R2)
489             *
490 050D4 D0            A=0    A              Clear nibs 3,4
491 050D6 15B2          A=DAT1 3              Read Text Table offset
492 050DA 172           D1=D1+ 3
493 050DD 147           C=DAT1 A              Read Rel. Execution address
494 050E0 133           AD1EX                 A <-- Pos in Table; D1 <- A
495 050E3 C2            C=C+A  A              Absolute Execution address
496 050E5 10A           R2=C                  Save Execution Address
497 050E8 133           AD1EX                 Restore A (Text Table offset)
498             *
499             * Compute Entry into Text Table
500             *    Entry Address = Text Table Offset + Text Table Start
501             *    Read # nibbles for Text
502             *    Read Text
503             *
```

```
504 050EB C0          A=A+B  A          Entry into Text Table
505 050ED 131         D1=A
506 050F0 1574        C=DAT1 S          # nibbles
507 050F4 170         D1=D1+ 1          Skip over # nibbles
508 050F7 1537        A=DAT1 W          Read text
509 050FB AC5         B=C    S          Preserve C(S)
510 050FE 11A         C=R2              Set D1 @ Execution Address
511 05101 135         D1=C
512 05104 AC9         C=B    S          Restore C(S)]
513 05107 03          RTNCC
514             *
```

```
515                     STITLE LINE#DC - Decompile Line#
516             *
517             *
518             ****************************************************************
519             ****************************************************************
520             **
521             ** Name:(S) LIN#DC  -  Line number decompile
522             ** Name:(S) LIN#D+  -  Line number decompile
523             ** Name:(S) LIN#AU  -  Line number decompile
524             ** Name:    LIN#A+  -  Line number decompile
525             ** Name:    LIN#CK  -  Line number decompile
526             **
527             ** Category:   DCMUTL
528             **
529             ** Purpose:    Decompiles a line number & outputs it
530             **
531             ** Entry:
532             **           P=0
533             **           D(A) points to end of available memory (AVMEME)
534             **           DO positioned at where decompiled line number to go
535             **             5 ENTRY POINTS :
536             **           1) LIN#CK  - Returns with carry set if A(B) # tLINE#
537             **                        Otherwise, falls into LIN#D+ entry pt.
538             **           2) LIN#D+  - Assumes D1 is 7 nibs prior to 1st
539             **                        digit of 4 nibble line number
540             **                        (2 nib line# token, 5 nib jump addr)
541             **                        Suppresses leading 0's
542             **           3) LIN#DC  - Same as above, except assumes that D1
543             **                        is pointing to 1st digit of line number
544             **           4) LIN#AU  - Used by TRACE - also suppresses leading
545             **                        zeroes.
546             **                        P=0 => 4 digits output, leading zeroes
547             **                               suppressed.
548             **                        P=12 => Convert from HEX to DEC. 2
549             **                               digits output, up to 4 leading
550             **                               zeroes suppressed.
551             **                        P=14 => Convert from HEX to DEC. 2
552             **                               digits output, up to 6 leading
553             **                               zeroes suppressed.
554             **
555             **           5) LIN#A+  - Used by System command AUTO - same as
556             **                        above except line# already in B(3-0)
557             **
558             ** Exit:       DO updated/ P=0/ Carry clear
559             **             LIN#AU, LIN#A+ - D1 left intact
560             **             LIN#DC, LIN#D+ - D1 stepped over 4 nibble line#
561             **
562             **
563             ** Calls:      DORSCI
564             **
565             ** Uses:       A, B, C, D(S), P
566             **
567             ** Stack lvls: 2
568             **
569             ** History:
```

```
570                **
571                **    Date     Programmer   Modifications
572                **  --------   ----------   -------------
573                **  07/06/82   S.W.         Added documentation
574                **  10/18/82   S.W.         Added P=0 entry condition
575                **
576                ********************************************************************
577                ********************************************************************
578 05109 3100   LIN#CK LC(2)  =tLINE#
579 0510D 966           ?A#C   B
580 05110 00            RTNYES
581                *
582 05112 176   =LIN#D+ D1=D1+ 7
583 05115 3133 =LIN#DC LCHEX  33              Read digits @ (D1); suppress
584 05119 7010         GOSUB  d0asci            leading zeroes; 4 digits.
585 0511D 173          D1=D1+ 4               Step over line#
586 05120 03           RTNCC
587                *
588 05122 AF8   =LIN#AU B=A    W
589 05125 353B =LIN#A+ LCHEX  D1F1B3          Find digits in B; suppress
          1F1D
590 0512D 8C00   d0asci GOLONG =DOASCI         leading zeroes
          00
591                *
592                *
593                ********************************************************************
594                ********************************************************************
595                **
596                ** Name:    ADDCHR  -  Add character
597                **
598                ** Category:   LOCAL
599                **
600                ** Purpose:
601                **      Builds variable token in B
602                **
603                ** Entry:
604                **      3 entry points:
605                **        1) ADCH++ -  A(B) contains next byte of token
606                **                     D1 pointing at that token
607                **        2) ADDCH+ -  C(B) already contains next token byte
608                **                     D1 as above
609                **        3) ADDCHR -  C(B) already contains next token byte
610                **                     D1 pointing at following token
611                **
612                ** Exit:
613                **      B shifted left twice and 'next token byte' in B(B)
614                **      C(S) incremented by 2
615                **      D1 pointing at next byte to examine & it's already been
616                **         read into A(B)
617                **
618                ** Calls:      AD1+2
619                **
620                ** Uses.......
621                **  Inclusive: A(B), B(W), C(B), C(S), D1
622                **
```

```
623             ** Stk lvls:   1
624             **
625             ** History:
626             **
627             **    Date      Programmer              Modification
628             **  --------    ----------   ------------------------------------
629             **  07/06/82    S.W.         Added documentation
630             **
631             ********************************************************************
632             ********************************************************************
633             *
634 05133 AE6    ADCH++ C=A     B
635 05136 7045   ADDCH+ GOSUB   AD1+2
636 0513A B46    ADDCHR C=C+1   S
637 0513D B46           C=C+1   S
638 05140 BF1           BSL     W
639 05143 BF1           BSL     W
640 05146 AE5           B=C     B
641 05149 01            RTN
```

```
                        STITLE ASCICK - Output ASCII chars
                *****************************************************************
                *****************************************************************
                **
                ** Name:(S) ASCICK  -  Ascii Stream Decompiler
                ** Name:    ASC02   -  Ascii Stream Decompiler
                **
                ** Category:   DCMUTL
                **
                ** Purpose:    Outputs stream of ascii characters
                **
                ** Entry:     3 ENTRY POINTS :
                **           D0 points to where output to go
                **           D(A) contains end of available memory (AVMEME)
                **           1) ASCI+  - D1 at 2 nibs prior to alleged start
                **                       of stream.
                **           2) ASCICK - D1 at start of alleged ascii stream.
                **           3) ASC02 -  Same as (2) above, only 1st character
                **                       already known to be ascii & is in C(B)
                **
                ** Exit:      Carry clr
                **            D1 past the ascii stream
                **            C(B) contains 'terminating' 1-byte token
                **
                ** Calls:     OUTBYT
                **
                ** Uses:      A(B), C(B), D1, D0
                **
                ** Detail:    If there's no ascii characters, nothing will be
                **            output & D1 will be left at 1st token
                **            Interprets as ascii any 1 byte token in which
                **            bit 7 is clear.
                **
                ** Stack lvls: 2
                **
                ** History:
                **
                **    Date      Programmer   Modifications
                **  --------    ----------   -------------
                **  07/06/82    S.W.         Improved documentation
                **
                *****************************************************************
                *****************************************************************
                *
0514B 171    ASCI+  D1=D1+ 2
0514E 14F   =ASCICK C=DAT1 B
05151 0B            CSTEX
05153 877           ?ST=1  7              NOT AN ASCII CHARACTER?
05156 B0            GOYES  ASC03
05158 0B            CSTEX
0515A 7285  =ASC02  GOSUB  Outbyt         WRITE OUT ASCII CHARACTER
0515E 5CE           GONC   ASCI+          (B.E.T.)
                *
05161 0B     ASC03  CSTEX
05163 03            RTNCC
```

```
697             *
```

```
                    STITLE ARYDC - Decompile Array
                *
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) ARYDC   -  Array Decompile
                **
                ** Category:   DCMUTL
                **
                ** Purpose:    Decompiles Array compiled in ARRYCK format
                **
                ** Entry       P= 0
                **      D(A) contains available memory end (AVMEME)
                **      D0 points into output buffer
                **      2 entry points:
                **      1) ARYDC  - Assumes C(B)=a(
                **                  D1 at first subscript, S5=0
                **      2) ARYDC+ - Checks for substring declaration (tSEMIC)
                **                  in A(B).  If not found, returns w/carry set.
                **                  Else D1 stepped over tSEMIC & expression
                **                  decompiled enclosed in brackets.
                **
                ** Exit:
                **       Carry clear=>
                **             subscripts output between parens (or brackets)
                **               parens (or brackets)
                **             D1 at token following last subscript
                **             A(B) contains the token
                **             If ARYDC+ called, S5=1
                **       Carry set (ARYDC+ entry only) =>
                **             No subscript decl. found
                **
                ** Calls:     OUTBYT, OBEXPR
                **
                ** Uses:      A-C, D1,D0, S5
                **            R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
                **
                ** Stack lvls: 5
                **
                ** History:
                **
                **    Date      Programmer   Modifications
                **  --------    ----------   -------------
                **  07/06/82    S.W.         Improved documentation
                **  08/16/82    S.W.         Added ARYDC+ entry
                **
                ********************************************************************
                ********************************************************************
                *
                *
05165 3100   ARYDC+ LC(2) =tSEMIC
05169 966           ?A#C   B
0516C 00            RTNYES
                *
0516E 171           D1=D1+ 2
```

```
753 05171 855          ST=1   5
754 05174 31B5         LCASC  \[\
755             *
756 05178 7F75 =ARYDC  GOSUB  OBEXPR         Outbyt/EXPRDC
757 0517C 3192         LCASC  \)\
758 05180 865          ?ST=0  5              DECOMPILING ARRAY?
759 05183 60           GOYES  ARYDC2
760 05185 31D5         LCASC  \]\
761 05189 7355  ARYDC2 GOSUB  Outbyt
762 0518D AEA          A=C    B              PUT TOKEN IN A(B)
763 05190 03           RTNCC
764             *
765             ********************************************************************
766             ********************************************************************
767             **
768             ** Name:(S) GTEXT+  -  GTEXT Preprocessor
769             ** Name:(S) GTXT++  -  GTEXT Preprocessor
770             ** Name:(S) GTEXT1  -  GTEXT Preprocessor
771             ** Name:(S) BLNKCK  -  Blank Check
772             **
773             ** Category:   DCMUTL
774             **
775             ** Purpose:
776             **      Given a keyword, GTEXT+, GTXT++, and GTEXT1 outputs
777             **      the corresponding text.
778             **
779             **      The BLNKCK entry point ensures that there is exactly
780             **      one blank after the last item decompiled.
781             **
782             ** Entry:
783             **      For all entry points:
784             **      P      =  0
785             **      D(A)   =  AVMEME
786             **      D0     =  Ptr to output buffer
787             **
788             **      BLNKCK entry:
789             **      -------------
790             **      No additional entry requirements
791             **
792             **      GTEXT+, GTXT++, GTXT1 entry:
793             **      ----------------------------
794             **      S9=1 => Output a trailing blank
795             **      D1      at keyword
796             **
797             **        1) GTXT++ - Outputs a leading & trailing blank
798             **                    Sets S9; Doesn't attempt to decompile
799             **                    text if token < 7E
800             **        2) GTEXT+ - Doesn't attempt to decompile text if
801             **                    token < 7E
802             **        3) GTEXT1 - Assumes A(B) already loaded with token
803             **                    greater than 6A.  No leading blank output
804             **
805             ** Note:  Can't call 1 or 2 above if want to output text
806             **         associated with a keyword in the range 6A-7D
807             **
```

```
                ** Exit:
                **      GTEXT+, GTXT++, GTEXT1 entry:
                **      ----------------------------
                **      P      =  0
                **      S9 set if GTXT++ used
                **      Carry set => Keyword not found, D1 intact
                **           clr => Text output, D1 past token, D0 advanced
                **
                **      BLNKCK entry:
                **      -------------
                **      Exactly 1 blank follows last item decompiled.
                **      D0 points past that blank
                **
                ** Calls:      GTEXTI, OUTBYT, OUTNBS
                **
                ** Uses:       A-C, R1-R2, D1,D0     (GTEXT1, GTEXT+ entry)
                **             A-C, R1-R2, D1,D0, S9 (GTXT++ entry)
                **             A(B),C(B),D0          (BLNKCK entry)
                **
                ** Stk lvls:   2  BLNKCK entry
                **             3  All other entry points
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ----------------------------------
                **  08/12/82   S.W.         Routine created
                **
                ********************************************************************
                ********************************************************************
                *
05192 859       =GTXT++ ST=1   9
05195 7820              GOSUB  GTXT+1
                *
05199 148       =GTEXT+ A=DAT1 B
0519C 31D7              LCHEX  7D
051A0 9EA               ?A<=C  B               Possible variable?
051A3 00                RTNYES
                *
051A5 7DCE      =GTEXT1 GOSUB  GTEXTI
051A9 4D2               GOC    GTXT+3          KEYWORD NOT FOUND?
051AC 80DF              P=C    15
051B0 7272              GOSUB  OUTNBS          OUTPUT ASCII
051B4 119               C=R1
051B7 135               D1=C                   RESTORE D1
051BA 879               ?ST=1  9               OUTPUT TRAILING BLANK?
051BD 40                GOYES  GTXT+1
051BF 03                RTNCC
                *
051C1           =BLNKCK
051C1 3102       GTXT+1 LCASC  \ \
051C5 181        BLNKC1 D0=D0- 2
051C8 14A               A=DAT0 B               Read in previous decompiled char
051CB 962               ?A=C   B               A blank?
051CE 7F                GOYES  BLNKC1
```

```
                * Found non-blank char
051D0 161       D0=D0+ 2
051D3 6C05      GOTO   Outbyt
                *
051D7 1C1   GTXT+3 D1=D1- 2             RESTORE D1
051DA 02        RTNSC
                *
                ******************************************************************
                ******************************************************************
                **
                ** Name:    FNCK    -  FN Check
                **
                ** Category:   DCMUTL
                **
                ** Purpose:
                **      Distinguishes between implied let of FN and a reference:
                **        FNX=15  vs  A(FNX)
                **
                ** Entry:
                **      P    =    0
                **      A(B) =    1st 1-byte token in statement
                **      D(A) =    Available Memory End
                **      D1        Points to supposed tFN
                **      D0        Points into output buffer
                **
                ** Exit:
                **      P      =  0
                **      D1,D0 unmoved from entry
                **      Carry set => Not implied LET
                **            clr => Is implied LET
                **
                ** Calls:      R3=D1O, VARDC, D1C=R3
                **
                ** Uses: A, B(A), C, R2, S6
                **
                ** Stk lvls:   2
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ----------------------------------
                **  08/29/83   S.W.         Added documentation header
                **
                ******************************************************************
                ******************************************************************
                *
051DC 3100 =FNCK   LC(2)  =tFN
051E0 966          ?A#C   B
051E3 00           RTNYES
051E5 11B          C=R3
051E8 10A          R2=C                 Save R3 in R2
051EB 8E00         GOSUBL =R3=D1O       Save D1 & D0 in R3
      00
051F1 7481         GOSUB  VARDC+        Step over tFN & compiled variable
051F5 1533         A=DAT1 X             Read in 3 nibbles past tFN tVAR
```

```
917 051F9 D8          B=A    A              Save token
918 051FB 8E00        GOSUBL =D1C=R3        Point D1 back to tFN
          00
919 05201 134         D0=C                  Restore D0
920 05204 11A         C=R2
921 05207 10B         R3=C                  Restore R3
922 0520A 32F1        LCHEX  F1F            Look for zero parm, and t=
          F
923 0520F 935         ?B#C   X              Distinguish between A(FNX)=..
924 05212 00          RTNYES                 & FNX=..
925 05214 01          RTN
926             *
```

```
                        STITLE Decompile User DEF declares
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:    DEFDC   -  DEF, FN Decompile
                    ** Name:    FNDC    -  DEF, FN Decompile
                    **
                    ** Category: STDCMP
                    **
                    ** Purpose:  DECOMPILES USER DEFINED FUNCTION DECLARATIONS
                    **
                    ** Entry:    D1 points to tFN
                    **           D0 points into output buffer
                    **           P=0
                    **           D(A) contains available memory end (AVMEME)
                    **
                    ** Exit:     via OUTELA
                    **
                    ** Calls:    DC=EXP, VARDC, OUT2TK, OUT1TK, ARYDC, OUTLP+, EOLXC*
                    **
                    ** Uses:     A-C, R0-R2, D1,D0, S0,S3,S6,S8,S10,S11
                    **
                    ** Stk lvls: 6
                    **
                    ** History:
                    **
                    **    Date     Programmer   Modifications
                    **  --------   ----------   ------------
                    **  08/18/82   S.W.         Added documentation
                    **  06/20/83   S.W.         Expanded test for tFN to
                    **                           not include A(FNX)=1
                    **  06/27/83   S.W.         Must preserve R3
                    **
                    ****************************************************************
                    ****************************************************************
                    *
                    *
05216 174   =DEFDC  D1=D1+ 5              STEP OVER 5 NIB FIELD
05219 3364  =FNDC   LCASC  \NF\           Output FN
      E4
0521F 77F5          GOSUB  out2tc
05223 7251          GOSUB  VARDC+         OUTPUT VARIABLE
05227 7A3F          GOSUB  ARYDC+         CK FOR SUBSTRING DECL
0522B 450           GOC    DEFDC1
0522E 171           D1=D1+ 2              STEP OVER CLOSING tSEMIC
05231 170   DEFDC1  D1=D1+ 1              STEP OVER PARM COUNT
05234 14B           A=DAT1 B
05237 3100          LC(2)  =tPRMST
0523B 966           ?A#C   B              NO PARAMETER LIST?
0523E 02            GOYES  DEFDC6
05240 7F74          GOSUB  Outlp+         WRITE OUT LEFT PAREN
05244 7431  DEFDC2  GOSUB  VARDC          OUTPUT PARAMETER
05248 3192          LCASC  \)\
0524C 962           ?A=C   B              AT END OF PARAMETER LIST?
0524F 90            GOYES  DEFDC4
```

```
981             * MUST BE ANOTHER PARM
982 05251 7784          GOSUB  Outby,          WRITE OUT COMMA
983 05255 5EE           GONC   DEFDC2          (B.E.T.)
984             *
985 05258 8E00  DEFDC4  GOSUBL =OUT1T+         OUTPUT RIGHT PAREN
          00
986 0525E 7A80  DEFDC6  GOSUB  EOLXC*          Doesn't return if stmt end
987             *
988 05262 561           GONC   LETDC3          Output '=' & Expression (B.E.T.)
989             *
990 05265 14B   COMTST  A=DAT1 B
991 05268 8C00  COM10   GOLONG =COMCK1
          00
992             *
```

```
                        STITLE LETDC - Decompile LET stmt
                ****************************************************************
                ****************************************************************
                **
                ** Name:    LETDC   -  LET Statement Decompile
                **
                ** Category:   STDCMP
                **
                ** Purpose:
                **      Decompiles LET Statement
                **
                ** Entry:
                **      D1 @ Assignment Variable or FN token
                **      D0 points into output buffer
                **      P= 0
                **      D(A) contains end of available memory (AVMEME)
                **      A(B) contains token pointed to by D1
                **
                ** Exit:
                **      Through FNDC if LET FN...
                **      Through OUTELA if LET <var>....
                **
                ** Calls:       EXPRDC, DC=EXP, FNCK
                **
                ** Uses:        A-C, R2, D1,D0
                **              R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
                **
                ** Stk lvls:   5
                **
                ** Detail:
                **      <variable> COMMA <expression>
                **
                ** Algorithm:
                **
                **      If FN token
                **         Set FN Decompile flag
                **         goto FN Decompile
                **      Decompile variable  (EXPRDC)
                ** 1:   Output "="          (OUTBYT)
                **      Decompile expr      (EXPRD+) (increments D1 @ entry)
                **      go Output terminating token (@ | ! | EOL)   (OUTELA)
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **   --------   ----------   ----------------------------------
                **   07/13/82   J.P.         Modified documentation
                **
                ****************************************************************
                ****************************************************************
                *
                *
 0526E 7A6F =LETDC  GOSUB  FNCK
 05272 56A          GONC   FNDC          Implied Let of FN ?
 05275 7000         GOSUB  =EXPRDC       Decompile variable
```

```
1048 05279 7774  LETDC3 GOSUB  DC=EXP
1049 0527D 504          GONC   outela        (B.E.T.) Output @ OR ! OR EOL
1050            *
```

```
1051                    STITLE DECDC - Decompile Declaration Stmts
1052                ****************************************************************
1053                ****************************************************************
1054                **
1055                ** Name:(S) DSTRDC  -  Decompiles Variable Declarations
1056                ** Name:    DECDC   -  Decompiles Variable Declarations
1057                **
1058                ** Category: STDCMP
1059                **
1060                ** Purpose:  Decompiles the following statements:
1061                **           INTEGER, SHORT, REAL, DIM, DESTROY, NEXT
1062                **
1063                ** Entry:    2 entry points:
1064                **           D(A) contains end of available memory
1065                **           P= 0
1066                **           D1 points into token stream
1067                **           D0 points into ascii output buffer
1068                **           1) DSTRDC - for statements with a possible
1069                **               keyword, eg TRACE and DESTROY.
1070                **           2) NXTDC
1071                **              DECDC  - For variable list, eg
1072                **               INTEGER, SHORT, REAL, DIM, NEXT
1073                **
1074                ** Exit:       A(B)=EOL TOKEN
1075                **             via OUTELA
1076                **
1077                ** Calls:    VARDC, ARYDC, OUTBYT, GTEXT+, EOLXC*
1078                **
1079                ** Uses:     A, C, S5,S6,S9, D1,D0
1080                **           A-C, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
1081                **
1082                ** Stk lvls: 6
1083                **
1084                ** History:
1085                **
1086                **    Date     Programmer   Modifications
1087                **  --------   ----------   -------------
1088                **  08/18/82   S.W.         Added documentation
1089                **
1090                ****************************************************************
1091                ****************************************************************
1092                *
1093                *
1094 05280 859  =DSTRDC ST=1   9              TRAILING BLANK
1095 05283 721F         GOSUB  GTEXT+
1096 05287 845  =DECDC  ST=0   5              STRING LENGTH FLAG
1097 0528A 7EE0 =NXTDC  GOSUB  VARDC
1098 0528E 866          ?ST=0  6              NOT AN ARRAY?
1099 05291 A0           GOYES  DECDC2
1100 05293 3182         LCASC  \(\
1101 05297 7DDE         GOSUB  ARYDC
1102                *D1 POINTING AT COMMA, EOL, OR SEMICOLON
1103 0529B 76CE  DECDC2 GOSUB  ARYDC+
1104                *
1105 0529F 7940         GOSUB  EOLXC*         Doesn't rtn if stmt end
```

```
1106             *
1107 052A3 7234          GOSUB  Outb,+
1108 052A7 5FD           GONC   DECDC          (B.E.T.)
1109             *
```

```
1110                    STITLE NXTDC - Decompile FOR/NEXT statement
1111                ****************************************************************
1112                ****************************************************************
1113                **
1114                ** Name:    FORDC   -  Decompiles FOR
1115                **
1116                ** Category: STDCMP
1117                **
1118                ** Purpose:  Decompiles FOR statement
1119                **
1120                ** Entry:    D1 POINTS TO INDEX VARIABLE
1121                **           D0 points into ascii output buffer
1122                **           P= 0
1123                **           D(A) contains end of available memory (AVMEME)
1124                **
1125                ** Calls:    VARDC, Exprd*, DC=EXP, EOLXC*
1126                **
1127                ** Uses:     A-C, D1,D0, S6
1128                **           A-C, R0-R2, D1,D0, S0,S3,S8,S10,S11 -- EXPRDC
1129                **
1130                ** Stk lvls: 6
1131                **
1132                ** History:
1133                **
1134                **    Date     Programmer   Modifications
1135                **  --------   ----------   -------------
1136                **  08/18/82   S.W.         Added documentation
1137                **
1138                ****************************************************************
1139                ****************************************************************
1140                *
1141                *
1142 052AA 71D0  =FORDC  GOSUB  VARDC1
1143 052AE 7244         GOSUB  DC=EXP
1144                * Output leading & trailing blanks around TO
1145                * And decompile following expression
1146 052B2 7464         GOSUB  Exprd*         Doesn't rtn if stmt end
1147 052B6 7230         GOSUB  EOLXC*
1148                * Output leading & trailing blanks around STEP
1149                * And decompile following expression
1150 052BA 7C54         GOSUB  Exprd*
1151 052BE 544   outela GONC   OUTELA         MUST BE AT EOL (B.E.T.)
```

```
1152                    STITLE IFDC - Decompile IF statement
1153                ****************************************************************
1154                ****************************************************************
1155                **
1156                ** Name:    IFDC    -  IF Decompiler
1157                **
1158                ** Category: STDCMP
1159                **
1160                ** Purpose:  Decompiles IF statement
1161                **
1162                ** Entry:    D1 POINTS AT NUMERIC EXPRESSION
1163                **           D0 Output pointer
1164                **           D(A) contains end of available mem (AVMEME)
1165                **           P=0
1166                **
1167                ** Exit:     2 CASES:
1168                **                    1) LINE# OR LABEL FOLLOWS 'THEN' =>
1169                **                       DECOMPILATION OF STATEMENT COMPLETE.
1170                **                    2) DOES GOLONG TO LDCIF
1171                **
1172                ** Calls:    EXPRDC, LIN#CK, LABLDC, COMTST, GTEXT+, FINDA,
1173                **           OUTBYT
1174                **
1175                ** Uses:     A-C, S9, D1,D0
1176                **           A-C, R0-R2, D1,D0, S0,S3,S8,S10,S11 - EXPRDC
1177                **
1178                ** Stk lvls: 5
1179                **
1180                ** History:
1181                **
1182                **    Date     Programmer   Modifications
1183                **  --------   ----------   -------------
1184                **  08/18/82   S.W.         Added documentation
1185                **  06/21/83   S.W.         THEN & ELSE now update INADDR
1186                **
1187                ****************************************************************
1188                ****************************************************************
1189                *
1190 052C1 7000 =IFDC   GOSUB  =EXPRDC
1191                * Output leading & trailing blanks around THEN
1192 052C5 79CE         GOSUB  GTXT++
1193 052C9 7395  IFDC1  GOSUB  finda+          Read 6 nibs into A first
1194 052CD 00           CON(2) =tLBLRF         LABEL Reference Token
1195 052CF F52          REL(3) RESTDC
1196 052D2 00           CON(2) =tLINE#
1197 052D4 A52          REL(3) RESTDC
1198 052D7 00           CON(2) =t!
1199 052D9 C60          REL(3) !DC
1200 052DC 00           CON(2) 0
1201                *
1202                * Must be tEXTIF
1203                * Step over tEXTIF & write out address of the stmt length
1204                *  byte which follows to INADDR
1205                *
1206 052DE 65FC  ldcm20 GOTO   LDCM20
```

```
1207            *
1208 052E2 859   ELSEDC ST=1   9              WANT TRAILING BUT NO LEADING BLK
1209 052E5 7CBE          GOSUB  GTEXT1
1210 052E9 5FD           GONC   IFDC1         (B.E.T.)
1211            *
```

```
1212                    STITLE OUTELA,END Decompile
1213            ****************************************************************
1214            ****************************************************************
1215            **
1216            ** Name:(S) OUTELA  -  Output End of Stmt Terminator From A
1217            ** Name:(S) OUTEL1  -  Exit for End of Stmt Decompile
1218            ** Name:(S) EOLXC*  -  Check for End of Stmt Decompile
1219            ** Name:(S) TRACDC  -  TRACE Statement Decompile
1220            ** Name:    REMDC   -  REMark or DATA Statement Decompile
1221            ** Name:    OUTEOL  -  Output End of Statement
1222            ** Name:    ENDDC   -  Decompile END Statement
1223            **
1224            ** Category:   DCMUTL
1225            **
1226            ** Purpose:  Entry points to handle end of statement decompile
1227            **           and misc statement decompile
1228            **
1229            ** Entry:
1230            **      P=0
1231            **      D(A) contains AVMEME
1232            **      D1 points into token stream
1233            **      D0 points  into ascii output buffer
1234            **           ENTRY POINTS:
1235            **             1) OUTELA - also STOPDC
1236            **                D1 at statement terminator (already read
1237            **                into A(B))
1238            **             2) OUTEL1 - End of statement decompile
1239            **                D1 at statement terminator
1240            **             3) EOLXC* - Doesn't return if D1 is at stmt
1241            **                end, else does
1242            **             4) TRACDC - TRACE and DEFAULT decompile
1243            **                         Outputs single keyword - no blanks
1244            **             5) REMDC  - also DATADC; D1 pointing after
1245            **                tREM or tDATA.
1246            **             6) OUTEOL - D1 at tEOL
1247            **             7) ENDDC  - Looks for ALL token
1248            **                         Falls into OUTEL1
1249            **
1250            ** Exit:
1251            **             If not called externally, exits via LSTLEN
1252            **              with carry clear
1253            **             If upon entry, D1 at tEOL or t! :
1254            **              D1 at tEOL, D1 untouched
1255            **                          D0 pts past last decompiled char
1256            **                          B(A)=#chars in buffer
1257            **
1258            **             If upon entry, D1 at tELSE or t@ :
1259            **              Decompile is continued, via ELSEDC &
1260            **              LDCM20, respectively.
1261            **
1262            **             If SST decompile and ELSE
1263            **                ELSE statement NOT decompile
1264            **                Jump to tEOL processing
1265            **
1266            **             If SST decompile and Multi-statement line
```

```
1267                **              Decompile does not continue
1268                **              Don't decompile past @
1269                **
1270                **            REM/DATA entry points - statement decompiled
1271                **
1272                ** Calls:    BLNKCK, OUT1TK, EOLDC, GTEXT1, !TEST, OUT2TK,
1273                **           TRNFCK, REMP10
1274                **
1275                ** Stack lvls: 4
1276                **
1277                ** Uses:     sSSTdc (S1)
1278                **
1279                ** History:
1280                **
1281                **    Date     Programmer   Modifications
1282                **  --------   ----------   -------------
1283                **  07/07/82   S.W.         Improved documentation
1284                **  08/30/82   J.P.         Added SST/ELSE checks
1285                **  10/27/82   J.P.         Added END ALL Decompile
1286                **
1287                ****************************************************************
1288                ****************************************************************
1289                *
1290 052EC 7211  =EOLXC* GOSUB  EOLDC
1291 052F0 500          RTNNC                  Carry always CLEAR
1292 052F3 07           C=RSTK                 Pop return stack
1293 052F5 4D0          GOC    OUTELA          (B.E.T.)
1294                *
1295 052F8 70FF  =ENDDC  GOSUB  EOLXC*         Check for EOL
1296 052FC       =TRACDC
1297 052FC 75AE  =DEFADC GOSUB  GTEXT1         Output ALL token
1298                *                          Fall into EOL processing
1299 05300 14B   =OUTEL1 A=DAT1 B
1300                *
1301 05303       =STOPDC
1302                *
1303 05303 3100  =OUTELA LC(2)  =tTHEN          THEN TOKEN
1304 05307 966          ?A#C   B               NOT @ ?
1305 0530A 03           GOYES  OUTEL3
1306 0530C 71BE         GOSUB  BLNKCK
1307 05310 173          D1=D1+ 4
1308 05313 14B          A=DAT1 B
1309                *
1310                * If ELSE and SST Decompile
1311                *    Avoid decompile ELSE clause
1312                *    Return
1313                *
1314 05316 3100          LC(2)  =tELSE
1315 0531A 966           ?A#C   B              A 'TRUE' @ ?
1316 0531D A0            GOYES  OUTEL4
1317 0531F 861           ?ST=0  sSSTdc         not SST Decompile ?
1318 05322 0C            GOYES  ELSEDC         Decompile ELSE clause
1319 05324 593           GONC   OUTEOL         B.E.T. - Don't decompile ELSE
1320                *
1321                * MULTI-LINE STMT
```

```
1322                *
1323 05327 1C3   OUTEL4 D1=D1- 4
1324 0532A 3104         LCASC  \@\
1325 0532E 7EA3         GOSUB  Outbyt
1326                *
1327                * If SST Decompile
1328                *    Return --> Don't decompile past "@"
1329                *
1330 05332 871          ?ST=1  sSSTdc          SST Decompile?
1331 05335 92           GOYES  OUTEOL          Return
1332 05337 56A          GONC   ldcm20          (B.E.T.)
1333                *
1334 0533A 78D0  OUTEL3 GOSUB  !TEST
1335 0533E 5F1          GONC   OUTEOL
1336                * COMMENT
1337 05341 7C7E         GOSUB  BLNKCK
1338 05345 3312  !DC    LCASC  \ !\            OUTPUT ! & LEADING BLANK
           02
1339 0534B 7BC4         GOSUB  out2tc
1340 0534F 171          D1=D1+ 2               Step over t!
1341                *
1342 05352      =DATADC
1343 05352 8E00 =REMDC  GOSUBL =REMP10         Decompile comment
           00
1344 05358 181          D0=D0- 2               Don't want OD output
1345                *
1346                * Reset AVMEMS back at Output Buffer Start
1347                *   Any decompile routine that POLLs has set AVMEMS @ Current D0
1348                *   Preserve D1 (only preserved if not externally invoked)
1349                *
1350 0535B 171          D1=D1+ 2               Step over OD
1351 0535E 171   OUTEOL D1=D1+ 2               Step over OF
1352 05361 133          AD1EX                  Preserve D1
1353 05364 8E00         GOSUBL =TRNFCK
           00
1354 0536A 460          GOC    DCDONE          Not called externally?
1355 0536D 06           RSTK=C                 Push return address on stack
1356 0536F 01           RTN
1357                * Ptr past OF in R0 is NOT valid if DC is externally invoked
1358 05371 D6    DCDONE C=A    A               Ptr past OF
1359 05373 8C00         GOLONG =LSTLEN         Save ptr past OF
           00
1360                *
```

```
1361                    STITLE VARDC - Decompile Variable
1362            ****************************************************************
1363            ****************************************************************
1364            **
1365            ** Name:(S) VARDC   -  Variable Decompile
1366            ** Name:    VARDC+  -  Variable Decompile
1367            **
1368            ** Category:   DCMUTL
1369            **
1370            ** Purpose:    Decompiles variables
1371            **
1372            ** Entry:
1373            **             P=0
1374            **             D(A) contains available memory end (AVMEME)
1375            **             D1   input pointer
1376            **             D0   output pointer
1377            **             S8=1 => no attempt to decompile arrays
1378            **                     (used by EXPRDC)
1379            **             2 entry points:
1380            **               1) VARDC+ - D1 2 nibs before alleged variable
1381            **               2) VARDC  - D1 at alleged variable
1382            **
1383            ** Exit:     P=0
1384            **             Regardless of S8:
1385            **             Carry clr => Variable found & decompiled
1386            **                          D1 past variable token
1387            **                          A(B)=B(B)= following token
1388            **
1389            **             S8 clr on entry:
1390            **             Carry clr => If S6 set, then decompiled
1391            **                          variable descriptor of array
1392            **             Carry set => no variable found
1393            **
1394            **             S8 set on entry:
1395            **             Carry clr => 00 byte output prior to decompiled
1396            **                          variable.
1397            **             Carry set => either variable not found or
1398            **                          encountered tARRAY
1399            **
1400            ** Calls:    ADDCHR, RANGE, OUTNBS
1401            **
1402            ** Uses:     A, B, C(S), C(A), S6
1403            **
1404            ** Stack lvls: 1
1405            **
1406            ** History:
1407            **
1408            **    Date     Programmer   Modifications
1409            **  ---------  ----------   -------------
1410            **  07/06/82   S.W.         Improved documentation
1411            **  10/18/82   S.W.         Added P=0 entry condition
1412            **  06/09/83   S.W.         Changed A=B B => A=B A (pack)
1413            **
1414            ****************************************************************
1415            ****************************************************************
```

```
1416                *
1417 05379 171   =VARDC+ D1=D1+ 2
1418 0537C 14B   =VARDC  A=DAT1 B
1419 0537F 846    VARDC1 ST=0   6               ARRAY FLAG
1420 05382 AC2           C=0    S               WP COUNT
1421 05385 3100          LC(2)  =tARRAY         Array token
1422 05389 966           ?A#C   B             NOT ARRAY TOKEN?
1423 0538C 01            GOYES  VARDC2
1424                *
1425 0538E 878           ?ST=1  8               CALLED BY EXPRDC?
1426 05391 00            RTNYES
1427 05393 856           ST=1   6               ARRAY TOKEN
1428 05396 171           D1=D1+ 2
1429 05399 14B           A=DAT1 B               GET TOKEN AFTER 'DO'
1430 0539C 3100  VARDC2  LC(2)  =tSVAR          STRING TOKEN
1431 053A0 966           ?A#C   B               NOT STRING TOKEN?
1432 053A3 A0            GOYES  VARDC4
1433 053A5 3142          LCASC  \$\
1434 053A9 798D          GOSUB  ADDCH+
1435 053AD 3306  VARDC4  LCHEX  6960            Range of alpha-digits
          96
1436 053B3 8E00          GOSUBL =Range          ALPHA-DIGIT TOKEN?
          00
1437 053B9 5E0           GONC   VARDC6
1438                *
1439                *    NOT ALPHA-DIGIT TOKEN
1440                *
1441 053BC 8E00          GOSUBL =ARANGE         ALPHA VAR TOKEN?
          00
1442 053C2 400           RTNC                   IF NOT, RETURN CARRY SET
1443 053C5 5D0           GONC   VARDC7          (B.E.T.)
1444                *
1445 053C8 3103  VARDC6  LCHEX  30
1446 053CC A86           C=A    P
1447 053CF 736D          GOSUB  ADDCH+          OUTPUT ALPHA-DIGIT TOKEN
1448 053D3 7C5D  VARDC7  GOSUB  ADCH++
1449 053D7 868           ?ST=0  8               NOT CALLED BY EXPRDC?
1450 053DA 90            GOYES  VARDC8
1451 053DC AE2           C=0    B
1452 053DF 775D          GOSUB  ADDCHR          OUTPUT NULL BYTE
1453 053E3 80FF  VARDC8  CPEX   15
1454 053E7 0D            P=P-1
1455 053E9 A9C           ABEX   WP
1456 053EC 7630          GOSUB  OUTNBS          WRITE OUT VAR TO INPUT BUF
1457 053F0 D4            A=B    A               Was 'A=B  B'
1458 053F2 03            RTNCC
1459                *
1460                *
```

```
1461                    $TITLE RDIZDC - Decompile RANDOMIZE Stmt
1462              ********************************************************************
1463              ********************************************************************
1464              **
1465              ** Name:    RDIZDC  -  RANDOMIZE Decompile
1466              **
1467              ** Category: STDCMP
1468              **
1469              ** Purpose:  DECOMPILES RANDOMIZE STATEMENT
1470              **
1471              ** Entry:    D1 POINTS past tRANDOMIZE
1472              **           D0 output pointer
1473              **           P= 0
1474              **           D(A) contains available memory end (AVMEME)
1475              **
1476              ** Exit:     via OUTELA
1477              **
1478              ** Calls:    OUT2TC, EXPRDC
1479              **
1480              ** Uses:     A-C, D1,D0
1481              **           A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
1482              **
1483              ** Detail:   RANDOMIZ[E] [<numeric expr>]
1484              **
1485              ********************************************************************
1486              ********************************************************************
1487              *
1488 053F4 3354 =RDIZDC LCASC  \ E\
          02
1489 053FA 7914         GOSUB  out2t0
1490 053FE 6170         GOTO   DROPDC
1491              *
1492              ********************************************************************
1493              ********************************************************************
1494              **
1495              ** Name:(S) EOLXCK  -  End of Stmt check
1496              ** Name:(S) EOLDC   -  End of Stmt check
1497              **
1498              ** Category:   GENUTL
1499              **
1500              ** Purpose:  Checks for statement terminator in the form of
1501              **             t! or t@ or tEOL
1502              **
1503              ** Entry:      P=0
1504              **             2 entry points:
1505              **               1) EOLDC  -  D1 at token in question
1506              **               2) EOLXCK -  A(B) contains token
1507              **
1508              ** Exit:     CARRY CLR=> No end of statement token found
1509              **
1510              ** Calls:      none
1511              **
1512              ** Stack lvls: 0
1513              **
1514              ** Uses:     C(B)
```

```
1515                **
1516                ** History:
1517                **
1518                **    Date     Programmer   Modifications
1519                **  --------   ----------   -------------
1520                **  07/07/82   S.W.         Improved documentation
1521                **  07/28/82   S.W.         Eliminated ELSE check
1522                **
1523                ************************************************************
1524                ************************************************************
1525                *
1526                *
1527 05402 14B  =EOLDC  A=DAT1 B
1528 05405 3100 =EOLXCK LC(2)  =tEOL
1529 05409 962          ?A=C   B
1530 0540C 00           RTNYES
1531 0540E 300          LC(1)  =t@              SAME AS tELSE, tTHEN
1532 05411 962          ?A=C   B
1533 05414 00           RTNYES
1534 05416 3100  !TEST  LC(2)  =t!              !
1535 0541A 962          ?A=C   B
1536 0541D 00           RTNYES
1537 0541F 01           RTN                     CARRY CLR
1538                *
1539                *
```

```
1540                    STITLE Output to Buffer
1541                ****************************************************************
1542                ****************************************************************
1543                **
1544                ** Name:(S) OUTNBS  -  Output nibbles
1545                ** Name:(S) OUTNBC  -  Output nibbles
1546                ** Name:(S) OUTC15  -  Output nibbles
1547                **
1548                ** Category:   GENUTL
1549                **
1550                ** Purpose:    Outputs specified number of nibbles from A or
1551                **             C to RAM pointed to by D0
1552                **
1553                ** Entry:      D(A) points to AVMEME
1554                **             D0 positioned properly
1555                **             3 entry points:
1556                **              1) OUTNBS - P set for WP write
1557                **                          Source in A
1558                **              2) OUTNBC - same as above except source in C
1559                **              3) OUTC15 - Outputs entire word from C
1560                **
1561                ** Exit:       P=0, Carry clear, D0 updated, D(A) preserved
1562                **
1563                ** Calls:      none
1564                **
1565                ** Stack lvls: 0
1566                **
1567                ** Uses:       C, A (all entry points except OUTNBS), P, D0
1568                **
1569                ** History:
1570                **
1571                **    Date     Programmer   Modifications
1572                **  --------   ----------   -------------
1573                **  07/08/82   S.W.         Improved documentation
1574                **  10/18/82   S.W.         Deleted OUTNC+, OUTNB+ entry
1575                **                           points; added OUTC15
1576                **
1577                ****************************************************************
1578                ****************************************************************
1579                *
1580                *
1581 05421 2F     =OUTC15 P=      15
1582 05423 A9A    =OUTNBC A=C     WP
1583 05426        =OUTNBS
1584 05426 136            CDOEX
1585 05429 134            D0=C
1586 0542C 809            C+P+1
1587 0542F 887            ?C>D    A
1588 05432 D0             GOYES   OUTOVF
1589 05434 1501           DAT0=A  WP
1590 05438 134            D0=C
1591 0543B 20             P=      0
1592 0543D 03             RTNCC
1593                *
1594                *
```

```
1595 0543F 8C00  OUTOVF GOLONG =MEMERR
           00
1596             *
1597             *
```

```
1598                    STITLE Decompile PRINT,FIX,DELAY,WAIT
1599            ****************************************************************
1600            ****************************************************************
1601            **
1602            ** Name:(S) PRNTDC  -  Expression List Decompile
1603            ** Name:(S) DISPDC  -  Expression List Decompile
1604            ** Name:(S) FIXDC   -  Expression List Decompile
1605            ** Name:(S) DROPDC  -  Expression List Decompile
1606            **
1607            ** Category: STDCMP
1608            **
1609            ** Purpose:  Decompiles PRINT, DISP, POKE, FIX, SCI, ENG, FLAG,
1610            **           DELAY, WAIT, INPUT, READ, statements
1611            **
1612            ** Entry:    P=0
1613            **           A(B) contains token pointed to by D1
1614            **           D(A) contains available memory end (AVMEM)
1615            **           D1   input pointer
1616            **           D0   output pointer
1617            **           PRNTDC - Entry FOR PRINT, DISP
1618            **             Allows USING to precede expression list
1619            **           FIXDC  - Entry FOR FIX, SCI, & ENG
1620            **             Must be at least 1 expression in list
1621            **           DROPDC - Entry for DROP, ADD
1622            **             Optional expression list (none necessary)
1623            **           INPTDC - Entry for INPUT
1624            **           READDC - Entry for READ, READ#
1625            **           SFLGDC - Entry for SFLAG, CFLAG
1626            **             Decompiles ALL, MATH, or expression list
1627            **
1628            ** Calls:    OUT1TK, EXPRDC, GTEXT+, EOLXC*, LIN#DC, -EXPR-,
1629            **           COMTST
1630            **
1631            ** Uses:     A-C, D1,D0, S9
1632            **           A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
1633            **
1634            ** Detail:   WILL WORK FOR ANY STATEMENT WHICH COMPILES TO A LIST
1635            **           OF EXPRESSIONS DELIMITED BY COMMA OR SEMI-COLON
1636            **           TOKENS.
1637            **           2 ENTRY POINTS:
1638            **             1) PRNTDC - FOR STATEMENTS WHICH OPTIONALLY ALLOW
1639            **                         A NULL LIST.
1640            **             2) DLAYDC - OTHERWISE
1641            **
1642            ** NOTE: tEND, tTAB, or \#\ MAY NOT BE USED AS A 'KLUDGE' TOKEN
1643            **       BY ANY ROUTINES THAT USE THIS ROUTINE.
1644            **
1645            ** Stk lvls: 6
1646            **
1647            ** History:
1648            **
1649            **    Date     Programmer   Modifications
1650            **  --------   ----------   -------------
1651            **  08/18/82   S.W.         Added documentation
1652            **
```

```
1653            ********************************************************************
1654            ********************************************************************
1655            *
1656 05445 96C  =INPTDC ?A#0   B              No prompt?
1657 05448 B4           GOYES  FIXDC
1658 0544A 171          D1=D1+ 2              Step over 0-byte preceding prompt
1659 0544D 554          GONC   FIXDC          (B.E.T.)
1660            *
1661 05450      =PRNTDC
1662 05450 3100 =DISPDC LC(2) =tUSING
1663 05454 966          ?A#C   B
1664 05457 91           GOYES  DROPDC
1665            *
1666 05459 859          ST=1   9              WANT TRAILING BLANK
1667 0545C 79F0         GOSUB  gtexta         OUTPUT 'USING '
1668 05460 75AC         GOSUB  LIN#CK         Decompile possible line#
1669 05464 6B00         GOTO   DROPDC
1670            *
1671 05468 31C2  PRTDC, LCASC  \,\
1672            *
1673 0546C 7752  PRTDC0 GOSUB  Outby+
1674            *
1675 05470      =DROPDC
1676 05470 787E =ADDDC  GOSUB  EOLXC*         MAY HAVE NO PARMS
1677            * NOT AN EOL TOKEN
1678 05474 70FD         GOSUB  COM10         COMMA Token low nibble
1679 05478 4FE          GOC    PRTDC,
1680 0547B E6           C=C+1  A
1681 0547D 962          ?A=C   B              SEMICOLON TOKEN?
1682 05480 A1           GOYES  PRTDC4
1683 05482 300          LC(1)  =tTAB          High nib also F
1684 05485 962          ?A=C   B              TAB TOKEN?
1685 05488 91           GOYES  PRTDC6
1686 0548A 3132 =READDC LCASC  \#\
1687 0548E 962          ?A=C   B
1688 05491 BD           GOYES  PRTDC0
1689
1690            *        LC(2)  =tEND
1691            *        ?A=C   B              END TOKEN? (FOR RESTORE#)
1692            *        GOYES  SFLGD5
1693            *
1694            * MUST BE AN EXPRESSION
1695            *
1696 05493      =LRDC
1697 05493      =WAITd
1698 05493      =WINDWd
1699 05493      =DELAYd
1700 05493 7000 =FIXDC  GOSUB  =EXPRDC
1701 05497 58D          GONC   DROPDC         (B.E.T.)
1702            *
1703            * SEMICOLON, #
1704            *
1705 0549A 31B3  PRTDC4 LCASC  \;\
1706 0549E 4DC          GOC    PRTDC0         (B.E.T.)
1707            *
```

```
1708            * TAB
1709            *
1710 054A1 700D  PRTDC6 GOSUB  GTEXT1          OUTPUT 'TAB' & STEP OVER TOKEN
1711 054A5 7422         GOSUB  -EXPR-
1712 054A9 AEA          A=C    B
1713 054AC 53C          GONC   DROPDC          (B.E.T.)
1714            *
1715 054AF 3100 =SFLGDC LC(2)  =tALL
1716 054B3 962          ?A=C   B
1717 054B6 F0           GOYES  SFLGD5
1718 054B8 35FE         LC(6)  =tMATH
           1063
1719 054C0 976          ?A#C   W
1720 054C3 0D           GOYES  FIXDC
1721 054C5 663E  SFLGD5 GOTO   TRACDC          Decompile last token - no blanks
1722            *
```

```
1723                    STITLE Decompile ASSIGN statement
1724                ********************************************************************
1725                ********************************************************************
1726                **
1727                ** Name:     ASSNDC  -  ASSIGN# and NAME Decompile
1728                ** Name:     NAMEDC  -  ASSIGN# and NAME Decompile
1729                **
1730                ** Category: STDCMP
1731                **
1732                ** Purpose:  Decompiles ASSIGN statement & NAME statement
1733                **           SYNTAX:
1734                **   ASSIGN# numeric expr TO <[filename]:dev addr | filename |
1735                **                           t*
1736                **
1737                ** Entry:    P= 0
1738                **           D(A) contains available memory end (AVMEME)
1739                **           D1   input pointer
1740                **           D0   output pointer
1741                **
1742                ** Exit:     P= 0
1743                **           via OUTELA
1744                **
1745                ** Calls:    GTEXT+, FILDC, ETMR20
1746                **
1747                ** Stk lvls: 6
1748                **
1749                ** Detail:   dev addr can be HP-IL addr, device word, or dev type
1750                **
1751                ** History:
1752                **
1753                **    Date      Programmer   Modifications
1754                **  --------    ----------   -------------
1755                **  08/18/82    S.W.         Added documentation
1756                **  10/18/82    S.W.         No longer decompile NULL,
1757                **                            DISPLAY
1758                **
1759                ********************************************************************
1760                ********************************************************************
1761                *
1762                *
1763 054C9 76A0 =ASSNDC GOSUB  ETMR20          OUTPUT '#' & CALL EXPRDC
1764                *
1765                * Output "TO"
1766                *
1767 054CD 71CC         GOSUB  GTXT++         Get "TO" text, output with blanks
1768                *
1769                * CHECK FOR t*, tNULL, tDISPLAY
1770 054D1 14B          A=DAT1 B
1771 054D4 3100         LC(2)  =t*
1772 054D8 966          ?A#C   B
1773 054DB A0           GOYES  NAMEDC
1774 054DD 31A2         LCASC  \*\
1775 054E1 6831         GOTO   KEYDC1         Output *; GOTO OUTEL1
1776                *
1777 054E5 7072 =NAMEDC GOSUB  FILDC*
```

```
1778 054E9 661E  ASND06 GOTO   OUTEL1
1779           *
```

```
1780                STITLE Decompile ON-GOTO,-GOSUB,-RESTORE
1781            ********************************************************************
1782            ********************************************************************
1783            **
1784            ** Name:    ONDC    -  ON..GOTO,..GOSUB,..RESTORE Decompile
1785            ** Name:(S) GOTODC  -  GOTO Decompile
1786            ** Name:(S) ONDC20  -  Keyword and Opt Line#/Label Decompile
1787            **
1788            ** Category:   STDCMP
1789            **
1790            ** Purpose:
1791            **      ONDC decompiles ON..GOTO,.. GOSUB,..RESTORE statements
1792            **
1793            **      GOTODC entry decompiles an optional list of line numbers/
1794            **      labels.  It is used by GOTO, GOSUB, and RESTORE decompile
1795            **      in the mainframe.
1796            **
1797            **      ONDC20 entry decompiles a keyword within leading and
1798            **      trailing blanks, then decompiles an optional list of line
1799            **      numbers/labels.
1800            **
1801            ** Entry:
1802            **      D(A) contains available memory end (AVMEME)
1803            **      D1   points into token stream
1804            **      D0   output pointer
1805            **      P= 0
1806            **      Entry points:
1807            **      ONDC   - D1 points to tERROR, tTIMER, or <expr>
1808            **      GOTODC - D1 points to start of optional list of
1809            **               line numbers/labels.
1810            **      ONDC20 - D1 points at keyword token preceding
1811            **               optional list of line numbers/labels
1812            **
1813            ** Exit:       Through PRNTDC
1814            **
1815            ** Calls:      EXPRDC,LIN#DC,LABLDC,OUTBYT,GTXT++,ETMRDC
1816            **
1817            ** Uses.......
1818            **  Exclusive: A-C, D1,D0,   S5,S9 (ONDC only)
1819            **  Inclusive: A-C, R0-R2, D1,D0, S0,S3,S8,S10,S11 - EXPRDC
1820            **
1821            ** Stk lvls:   6
1822            **
1823            ** Detail:
1824            **      ON ERROR (GOTO|GOSUB) (<lineno> | <label> )
1825            **      ON TIMER  #<timer no>,  <#secs> (GOTO|GOSUB)
1826            **               (<lineno> | <label>)
1827            **      ON <exp> GOTO    <lineno>|<label> [,<lineno>|<label>]
1828            **               GOSUB
1829            **               RESTORE
1830            **
1831            ** History:
1832            **
1833            **    Date     Programmer             Modification
1834            **  --------   ----------   ----------------------------------
```

```
1835            ** 07/13/82   J.P.          Modified documentation
1836            ** 08/29/83   S.W.          Updated documentation
1837            **
1838            ************************************************************************
1839            ************************************************************************
1840            *
```

```
1841                    STITLE RUN Decompile
1842                ********************************************************************
1843                ********************************************************************
1844                **
1845                ** Name:    RUNDC   -  Decompile RUN statement
1846                **
1847                ** Category:   STDCMP
1848                **
1849                ** Purpose:
1850                **      Decompile start of RUN statement
1851                **
1852                ** Entry:
1853                **      D1 past RUN token
1854                **      D0 output pointer
1855                **      D(A) contains available memory end (AVMEME)
1856                **      P= 0
1857                **
1858                ** Exit:
1859                **      via OUTELA
1860                **
1861                ** Calls:      FILDC+, COMTST, OUTBYT, LIN#CK, LABLDC
1862                **
1863                ** Uses.......
1864                **  Exclusive: A-C, D1,D0, sRUNDC (S7)
1865                **  Inclusive: A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
1866                **
1867                ** Stk lvls:   5
1868                **
1869                ** Detail:
1870                **      Tokenized forms:
1871                **
1872                **      tRUN tRFILE <file spec>
1873                **           [ tCOMMA tLINE# <jump addr> <line#> |
1874                **             tCOMMA tLBLRF <string expression> |
1875                **             tCOMMA tLBLRF tLITRL <ASCII label> ]
1876                **      tRUN tCOMMA tLINE# <jump addr> <line#>
1877                **      tRUN tCOMMA tLBLRF <string expression> |
1878                **           tCOMMA tLBLRF tLITRL <ASCII label> ]
1879                **      tRUN tLINE# ....
1880                **      tRUN
1881                **
1882                **      Decompile forms:
1883                **
1884                **      RUN <filespec> [ , <lineno> | <label> ]
1885                **      RUN <lineno>
1886                **      RUN , <label> | <lineno>
1887                **      RUN
1888                **
1889                ** History:
1890                **
1891                **    Date     Programmer            Modification
1892                **  --------   ----------   ----------------------------------
1893                **  07/13/82   J.P.         Modified documentation
1894                **
1895                ********************************************************************
```

```
1896                ****************************************************************
1897 054ED 7C50 =ONDC   GOSUB  ETMRDC          Decompile ERROR | TIMER
1898 054F1 5F0          GONC   ONDC20          ON ERROR statement
1899 054F4 865          ?ST=0  5               Not ON TIMER# <expr>,
1900 054F7 60           GOYES  ONDC10
1901 054F9 7CD1         GOSUB  Outb,+
1902                *
1903                * Decompile expression of ON TIMER #<expr> | ON <expr>
1904                *
1905 054FD 7000  ONDC10 GOSUB  =EXPRDC
1906                * Output leading & trailing blanks
1907 05501 7D8C =ONDC20 GOSUB  GTXT++          BLANKS
1908 05505 582          GONC   RESTDC          (B.E.T.)
1909                *
1910                * RUN Decompile Entry
1911                *     Ck. for Comma after <filespec> or before <lineno|label>
1912                *     If Not Comma and RUN Decompile
1913                *        go Check for <lineno>
1914                *
1915 05508 857  =RUNDC  ST=1   sRUNDC          Set RUN Decompile flag
1916 0550B 3100         LC(2)  =tRFILE
1917 0550F 966          ?A#C   B               No RUN file ?
1918 05512 60           GOYES  RUNDC+          Decompile [ <lineno> | <label> ]
1919 05514 7E32         GOSUB  FILDC+          Skip tRFILE, Decompile <filespec>
1920                *
1921 05518       RUNDC+
1922 05518 794D  ONDC40 GOSUB  COMTST
1923 0551C 4A0          GOC    ONDC42          Output it and continue
1924 0551F 877          ?ST=1  sRUNDC          Run Decompile ?
1925 05522 F0           GOYES  ONDC50          No comma, Look for Line#
1926 05524 54C          GONC   ASNDO6          B.E.T. - Must be stmt end
1927 05527 7EA1  ONDC42 GOSUB  Outb,+
1928 0552B 550          GONC   ONDC50          B.E.T. to Skip flag clearing
1929                *
1930                *
1931 0552E       =GOTODC
1932 0552E       =GOSBDC
1933 0552E 847   =RESTDC ST=0   sRUNDC         Clear RUN Decompile flag
1934 05531 14B   ONDC50 A=DAT1 B
1935 05534 71DB         GOSUB  LIN#CK          Decompile possible line#
1936 05538 5FD          GONC   ONDC40          Line# found ?
1937                *
1938                * Decompile Label
1939                *
1940 0553B CE           C=C-1  A               Label Reference Token
1941 0553D 966          ?A#C   B               NOT LABEL?
1942 05540 90           GOYES  ONDC80
1943 05542 7CB1         GOSUB  LABLDC          Decompile Label
1944 05546 51D          GONC   ONDC40          (B.E.T.)
1945                *
1946                * Not <lineno> | <label>
1947                *
1948 05549 662F  ONDC80 GOTO   DROPDC
1949                *
1950                *
```

```
1951                    STITLE Decompile ERROR/TIMER of ON/OFF
1952                ********************************************************************
1953                ********************************************************************
1954                **
1955                ** Name:    ETMRDC  -  ERROR/TIMER Decompile of ON/OFF
1956                **
1957                ** Category:   STDCMP
1958                **
1959                ** Purpose:
1960                **      Decompile ERROR | TIMER portion of ON & OFF statement
1961                **
1962                ** Entry:
1963                **      P= 0
1964                **      D(A) contains available memory end (AVMEME)
1965                **      D0 output pointer
1966                **      D1 past ON | ERROR token
1967                **      A(B) contains following token
1968                **
1969                ** Exit:
1970                **      Carry clear: ERROR
1971                **      Carry set:   TIMER | Not TIMER or ERROR
1972                **                   S5=1 if TIMER
1973                **
1974                **      A holds next token
1975                **
1976                ** Calls:      GTEXT+, OBEXPR
1977                **
1978                ** Uses.......
1979                **  Exclusive: A-C, R1,R2, S5,S9, D1,D0
1980                **  Inclusive: A-C, R0-R2, S0,S3,S8,S10,S11, D1,D0 - EXPRDC
1981                **
1982                **      S5=1 if TIMER #<expr>
1983                **
1984                ** Stk lvls:   5
1985                **
1986                ** Algorithm:
1987                **
1988                **      Clear TIMER# flag (S5)
1989                **      If next token = ERROR
1990                **         Output "ERROR "  (OUTNBC)
1991                **         Skip token, Read next token
1992                **         RTNCC
1993                **      If next token = TIMER
1994                **         Output "TIMER "  (OUTNBC)
1995                **         Output "#"       (OUTBYT)
1996                **         Set ON TIMER flag
1997                **         Decompile TIMER# <expr>  (EXPRDC)
1998                **         Read next token
1999                **      RTNSC
2000                **
2001                ** History:
2002                **
2003                **    Date     Programmer             Modification
2004                **  --------   ----------   ------------------------------------
2005                **  07/13/82   J.P.         Modified documentation
```

```
2006            **  07/07/83   S.W.         Eliminated A=DAT1 B as 2nd statement.
2007            **
2008            ************************************************************************
2009            ************************************************************************
2010            *
2011 0554D 845  =ETMRDC ST=0   5              Clear TIMER# flag
2012 05550 3100         LC(2)  =tERROR
2013 05554 966          ?A#C   B              not ERROR ?
2014 05557 B0           GOYES  ETMR10
2015            *
2016            * ERROR
2017            *
2018            * Assume S9 clear
2019 05559 7C3C  gtexta GOSUB  GTEXT+
2020 0555D 14B          A=DAT1 B              Read next token
2021 05560 03           RTNCC
2022            *
2023            * Not ERROR or TIMER
2024            *
2025 05562 E6    ETMR10 C=C+1  A              TIMER token
2026 05564 966          ?A#C   B
2027 05567 00           RTNYES                Not TIMER or ERROR
2028            *
2029            * TIMER # <expr>
2030            *
2031 05569 859          ST=1   9
2032 0556C 753C         GOSUB  GTEXT1
2033 05570 855          ST=1   5              Set TIMER# flag
2034 05573 3132  ETMR20 LCASC  \#\
2035 05577 7081         GOSUB  OBEXPR         Outbyt/EXPRDC
2036 0557B 02           RTNSC
```

```
                        STITLE OFFDC - Decompile OFF stmt
                *****************************************************************
                *****************************************************************
                **
                ** Name:    OFFDC   -  OFF Statement Decompile
                **
                ** Category:   STDCMP
                **
                ** Purpose:
                **      Decompile OFF statement
                **
                ** Entry:
                **      D1 past OFF
                **      D0 output pointer
                **      D(A) contains available memory end (AVMEME)
                **      P=0
                **
                ** Exit:
                **      Through OUTELA
                **
                ** Calls:      ETMRDC
                **
                ** Uses.......
                **  Exclusive: A-C, D1,D0, R1,R2, S5,S9
                **  Inclusive: A-C, R0-R2, D1,D0, S0,S3,S8,S10,S11 - EXPRDC
                **
                ** Stk lvls:   6
                **
                ** Detail:
                **      OFF [ ERROR | TIMER # <timer no>
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ---------------------------------
                **  07/13/82   J.P.         Modified documentation
                **
                *****************************************************************
                *****************************************************************
                *
05577 7CCF =OFFDC  GOSUB  ETMRDC          Decompile ERROR | TIMER
05581 618D         GOTO   OUTELA          Process EOL
```

```
2079                    $TITLE OPTDC - Decompile OPTION stmt
2080              ****************************************************************
2081              ****************************************************************
2082              **
2083              ** Name:    OPTDC   -  OPTION Decompile
2084              **
2085              ** Category: STDCMP
2086              **
2087              ** Purpose:  Decompiles OPTION, DEFAULT, TRACE statements
2088              **
2089              ** Entry:    D1 past tOPTION, tDEFAULT
2090              **           D0 output pointer
2091              **           D(A) contains available memory end (AVMEME)
2092              **           P= 0
2093              **           A(B) contains byte pointed to by D1
2094              **
2095              ** Calls:    GTEXT1, EXPRDC
2096              **
2097              ** Uses:     A-C, R1,R2, S8,S9, D1,D0
2098              **           A-C, R0-R2, D1,D0, S0,S3,S8,S10,S11 - EXPRDC
2099              **
2100              ** Stk lvls: 5
2101              **
2102              ** Detail:   OPTION BASE  <num exp>
2103              **           OPTION ANGLE  DEGREES | RADIANS
2104              **           OPTION ROUND  NEAR | PINF | NINF | ZERO
2105              **           DEFAULT ON | OFF | EXTEND
2106              **           TRACE   FLOW | VARS | OFF
2107              **
2108              ** History:
2109              **
2110              **    Date     Programmer   Modifications
2111              **  --------   ----------   -------------
2112              **  08/18/82   S.W.         Added documentation
2113              **
2114              ****************************************************************
2115              ****************************************************************
2116              *
2117              *
2118 05585 3100 =OPTDC  LC(2)  =tBASE
2119 05589 966          ?A#C   B
2120 0558C 50           GOYES  OPTDC1
2121 0558E 858          ST=1   8               BASE FLAG
2122 05591 859   OPTDC1 ST=1   9               WANT TRAILING BLANK
2123 05594 7D0C         GOSUB  GTEXT1
2124 05598 878          ?ST=1  8               BASE?
2125 0559B 90           GOYES  OPTDC2          YES=> DECOMPILE NUMERIC EXPR
2126 0559D 14B          A=DAT1 B               Read DEGREES or RADIANS token
2127 055A0 6B5D         GOTO   TRACDC
2128              *
2129 055A4 6EEE  OPTDC2 GOTO   FIXDC
2130              *
```

```
2131                    STITLE END SUB Decompile
2132                ********************************************************************
2133                ********************************************************************
2134                **
2135                ** Name:    ENDSDC  -  END SUB, END DEF Decompile
2136                **
2137                ** Category: STDCMP
2138                **
2139                ** Purpose:  Decompile END SUB & END DEF statements
2140                **
2141                ** Entry:    P= 0
2142                **           D(A) contains available memory end (AVMEME)
2143                **           D1 points past token
2144                **           D0 output pointer
2145                **
2146                ** Exit:     via OUTEL1
2147                **
2148                ** Calls:    OUTC15
2149                **
2150                ** Stk lvls: 1
2151                **
2152                ** Uses:     A,C, D1,D0
2153                **
2154                ** Detail:   Backs up over <= that has been output then outputs
2155                **           END SUB or END DEF
2156                **
2157                ********************************************************************
2158                ********************************************************************
2159                *
2160 055A8 187      =ENDSDC D0=D0- 8              Back up over 3 chars & blank
2161                                              which has been output
2162 055AB 3F54             LCASC  \ BUS DNE\     END SUB
           E444
           0235
           5524
           02
2163 055BD 706E   ENDS10    GOSUB  OUTC15        Output entire word
2164 055C1 174              D1=D1+ 5             Skip over link field
2165 055C4 6B3D   ENDS11    GOTO   OUTEL1
2166
2167 055C8 187      =ENDDDC D0=D0- 8              Back up over 3 chars & blank
2168                                              which has been output
2169 055CB 3F54             LCASC  \ FED DNE\     END DEF
           E444
           0244
           5464
           02
2170 055DD 5FD              GONC   ENDS10        (B.E.T.)
2171                *
2172                *
2173                *
```

```
2174                    STITLE KEYDC - Decompile KEY stmt
2175                ******************************************************************
2176                ******************************************************************
2177                **
2178                ** Name:    KEYDC   -  KEY statement decompile
2179                **
2180                ** Category: STDCMP
2181                **
2182                ** Purpose:  Decompiles DEF KEY statement
2183                **
2184                ** Entry:    D1 points past tKEY
2185                **           DO output pointer
2186                **           D(A) contains available memory end (AVMEME)
2187                **           P= 0
2188                **
2189                ** Exit:     via OUTEL1
2190                **
2191                ** Calls:    OUT1TK, OUTNBC, EXPRDC, EOLXC*
2192                **
2193                ** Uses:     A-C, D1,DO
2194                **           A-C, RO-R2, D1,DO, S0,S3,S8,S10,S11 -- EXPRDC
2195                **
2196                ** Stk lvls: 5
2197                **
2198                ** Detail:   WRITES OUT 'DEF KEY' IN PLACE OF 'KEY'
2199                **
2200                ** History:
2201                **
2202                **   Date      Programmer          Modification
2203                ** --------   ----------   ------------------------------
2204                ** 08/29/83    S.W.         Added documentation
2205                **
2206                ******************************************************************
2207                ******************************************************************
2208                *
2209 055E0 187     =KEYDC  DO=DO- 8              Back up over 'KEY '
2210 055E3 3F44            LCASC  \ YEK FED\
           5464
           02B4
           5495
           02
2211 055F5 782E            GOSUB  OUTC15
2212 055F9 7000            GOSUB  =EXPRDC        Decompile strng or num expr
2213 055FD 7BEC            GOSUB  EOLXC*         Doesn't rtn if stmt end
2214 05601 74D0            GOSUB  Outb,+         Write out comma; skip over token
2215 05605 7000            GOSUB  =EXPRDC        Decompile strng expr
2216                * Can only be stmt end (EOL,@,OR '), colon, or semi-colon
2217 05609 31B3            LCASC  \;\
2218 0560D 962             ?A=C   B              SEMI-COLON?
2219 05610 A0              GOYES  KEYDC1
2220                *
2221 05612 30A             LC(1)  \:\
2222 05615 966             ?A#C   B              NOT COLON?
2223 05618 CA              GOYES  ENDS11
2224 0561A 79A0  KEYDC1    GOSUB  Outby+
```

```
2225 0561E 55A          GONC   ENDS11         (B.E.T.)
2226            *
```

```
2227                    STITLE TRANSFORM Decompile
2228                ****************************************************************
2229                ****************************************************************
2230                **
2231                ** Name:    TRSFMD  -  TRANSFORM Decompile
2232                **
2233                ** Category:   STDCMP
2234                **
2235                ** Purpose:
2236                **      Decompile TRANSFORM statement.
2237                **
2238                ** Entry:
2239                **      P      =  0
2240                **      D0 past "TRANSFOR "
2241                **      D1 past TRANSFORM token
2242                **      D(A) contains available memory end (AVMEME)
2243                **
2244                ** Exit:
2245                **      Through CREADC & PRNTDC
2246                **      P      =  0
2247                **
2248                ** Calls:      FTYPDC, OUT2TC, FILDC, BLNKCK, GTEXT+, EOLXC*
2249                **
2250                ** Uses.......
2251                **  Exclusive: A-C, D1,D0, R0-R2, S8,S9
2252                **  Inclusive: A-C, R0-R2, D1,D0, S0,S3,S8,S10,S11 -- EXPRDC
2253                **
2254                ** Stk lvls:   6
2255                **
2256                ** Detail:     Must immediately precede CREADC
2257                **
2258                ** History:
2259                **
2260                **    Date     Programmer             Modification
2261                **  --------   ----------   ---------------------------------
2262                **  06/14/82   F.H.         Designed and coded.
2263                **  08/13/82   F.H.         Packed using SW's table routine
2264                **  08/29/83   S.W.         Updated documentation.
2265                **
2266                ****************************************************************
2267                ****************************************************************
2268 05621 33D4   =TRSFMD LCASC  \ M\           Output "M " of TRANSFOR[M]
           02
2269 05627 7CE1         GOSUB  out2t0           .
2270 0562B AF6          C=A    W                .
2271 0562E 15B5         A=DAT1 6
2272 05632 35FE         LC(6)  =tINTO           .
           10E2
2273 0563A 972          ?A=C   W                INTO?
2274 0563D A0           GOYES  TFMD20
2275 0563F 7611         GOSUB  FILDC*           Decompile file spec
2276 05643 7A7B         GOSUB  BLNKCK           Output blank
2277 05647 859    TFMD20 ST=1   9               Decompile "INTO" token
2278 0564A 7B4B         GOSUB  GTEXT+           .
2279                *
```

```
2280                *
```

```
2281                    STITLE CREATE Decompile
2282                ********************************************************************
2283                ********************************************************************
2284                **
2285                ** Name:    CREADC  -  CREATE Decompile
2286                **
2287                ** Category:   STDCMP
2288                **
2289                ** Purpose:
2290                **      Decompiles File type & following optional file spec
2291                **      & expressions
2292                **
2293                ** Entry:
2294                **      P       =  0
2295                **      D1 at file type#
2296                **      D0 output pointer
2297                **      D(A) contains available memory end (AVMEME)
2298                **
2299                ** Exit:
2300                **      P       =  0
2301                **      File type & file spec (if any) decompiled
2302                **
2303                ** Calls:      FTYPDC, EOLXC*, BLNKCK, FILDC
2304                **
2305                ** Uses.......
2306                **  Exclusive: A-C, D1,D0, R0
2307                **  Inclusive: A-C, R0-R2, D1,D0, S0,S3,S8,S10,S11 - EXPRDC
2308                **
2309                ** Stk lvls:   6
2310                **
2311                ** History:
2312                **
2313                **    Date      Programmer              Modification
2314                **  --------    ----------   ----------------------------------
2315                **  08/16/82    S.W.         Wrote for usage by CREADC, TRSFMD
2316                **
2317                ********************************************************************
2318                ********************************************************************
2319                *
2320                *
2321 0564E 8E00 00  =CREADC GOSUBL =FTYPD+        CK MEM & DECOMPILE FILE TYPE
2322                *
2323 05654 796B             GOSUB  BLNKCK        Ensure only 1 blank output
2324 05658 173              D1=D1+ 4             STEP OVER TYPE #
2325 0565B 7D8C             GOSUB  EOLXC*        No rtn if at stmt end
2326                * SINGLE BLANK OUTPUT
2327 0565F 76F0             GOSUB  FILDC*
2328 05663 6C0E             GOTO   DROPDC        Optional numeric expressions
2329                *
```

```
2330                    STITLE SUB Parm List Decompile
2331                ********************************************************************
2332                ********************************************************************
2333                **
2334                ** Name:     PARMCK  -  Parameter Check
2335                **
2336                ** Category: LOCAL
2337                **
2338                ** Purpose:  DECOMPILES SUB-PROGRAM NAME & CHECKS FOR PARM LIST
2339                **
2340                ** Entry:      P= 0
2341                **             D(A) contains available memory end (AVMEME)
2342                **             D1 points into token stream
2343                **             D0 points into ascii output buffer
2344                **             3 entry points:
2345                **             1) PRMCK+ - Decompiles sub-program name & looks at
2346                **                         parm list (see below) - for CALLDC
2347                **             1) PARMCK - Decompiles subprogram name & looks at
2348                **                         parm list - for SUBDC
2349                **             2) PRMCK3 - Returns with carry clear if parm list
2350                **                         done.  Decompiles parm list until a
2351                **                         parm unlike <#expr> is encountered.
2352                **
2353                ** Exit:     CARRY SET => NOT YET AT END OF STATEMENT
2354                **                  CLR => AT END OF STATEMENT - ) OUTPUT IF
2355                **                         APPROPRIATE.  D1 past tPRMEN
2356                **
2357                ** Calls:    FILDC, EXPRDC, OUTBYT, COMTST
2358                **
2359                ** Uses:     A-C, D1,D0, S8,S9
2360                **           A-C, R0-R2, D1,D0, S0,S3,S8,S10,S11 - EXPRDC
2361                **
2362                ** Detail:   Used by SUB and CALL decompile
2363                **
2364                ** Stk lvls: 6
2365                **
2366                ** History:
2367                **
2368                **    Date     Programmer   Modification
2369                **  --------   ----------   ----------
2370                **  07/07/83   S.W.         When subprgrm name isn't a literal
2371                **                           decompile it in parentheses.
2372                **
2373                ********************************************************************
2374                ********************************************************************
2375                *
2376                *
2377 05667 859     PRMCK+ ST=1   9              CALLDC Flag
2378                *
2379 0566A 7BE0    PARMCK GOSUB  FILDC*
2380 0566E 14B             A=DAT1 B
2381 05671 3100            LC(2)  =tPRMST       LEFT PAREN TOKEN
2382 05675 962             ?A=C   B             PARM LIST?
2383 05678 A0              GOYES  PRMCK1
2384                *
```

```
2385 0567A 171   AD1+2  D1=D1+ 2              STEP OVER tPRMEN
2386 0567D 14B          A=DAT1 B
2387 05680 03           RTNCC
2388            *
2389 05682 7D30  PRMCK1 GOSUB  Outlp+
2390 05686 14B          A=DAT1 B
2391 05689 5A1          GONC   PRMCK5          (B.E.T.)
2392            *
2393 0568C 14B   PRMCK3 A=DAT1 B
2394 0568F 3100         LC(2)  =tPRMEN
2395 05693 966          ?A#C   B               Not at end of parm list?
2396 05696 80           GOYES  PRMCK4
2397 05698 171          D1=D1+ 2               Step over tPRMEN
2398 0569B 593          GONC   Outrtp          (B.E.T.)
2399            *
2400 0569E 7A30  PRMCK4 GOSUB  Outby,
2401 056A2 DA           A=C    A
2402 056A4 3132  PRMCK5 LCASC  \#\
2403 056A8 966          ?A#C   B
2404 056AB 00           RTNYES
2405            *
2406 056AD 7740         GOSUB  OB+EXD          Step over #, Output #, Exprdc
2407            * If SUB, step over tCOMMA or tPRMEN
2408            * If CALL, step over 2 nibble parm descripter
2409            *
2410 056B1 171   PRMCK7 D1=D1+ 2
2411 056B4 879          ?ST=1  9               Not SUB decompile?
2412 056B7 5D           GOYES  PRMCK3
2413            * D1 past tCOMMA or tPRMEN
2414 056B9 7BAB         GOSUB  COM10
2415 056BD 571          GONC   Outrtp          Not comma?
2416            * D1 past tCOMMA; Read in next token
2417 056C0 4BC          GOC    PRMCK3          (B.E.T.)
2418            *
2419 056C3 3182  Outlp+ LCASC  \(\
2420 056C7 8C00  Outby+ GOLONG =OUTBY+         STEP OVER tPRMEN
         00
2421            *
2422            *
2423            ****************************************************************
2424            ****************************************************************
2425            **
2426            ** Name:    -EXPR-  -  Num Expr w/opt Parens Decompiler
2427            **
2428            ** Category:   DCMUTL
2429            **
2430            ** Purpose:
2431            **      Decompiles expression in parentheses
2432            **
2433            ** Entry:
2434            **      P      =  0
2435            **      D1 at start of Expression
2436            **      D0,D(A) as required by OUTBYT
2437            **      2 entry points:
2438            **      1) -EXPR- - Outputs left paren, expression, & right paren
```

```
2439            **      2) ?EXPR- - Outputs ascii char in C(B), expression &
2440            **                  right paren.
2441            **
2442            ** Exit:
2443            **      P       =  0
2444            **      Carry clear
2445            **      through OUTBYT
2446            **
2447            ** Calls:       OUTBYT, EXPRDC
2448            **
2449            ** Uses.......
2450            **  Exclusive: A,C, D1,D0
2451            **  Inclusive: A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 - EXPRDC
2452            **
2453            ** Stk lvls:   5
2454            **
2455            **
2456            ** History:
2457            **
2458            **    Date      Programmer              Modification
2459            **  --------    ----------   ----------------------------------
2460            **  08/16/82    S.W.         Wrote routine
2461            **
2462            ********************************************************************
2463            ********************************************************************
2464            *
2465 056CD 3182  -EXPR- LCASC  \(\
2466 056D1 7620  ?EXPR- GOSUB  OBEXPR          Outbyt/EXPRDC
2467 056D5 6000  Outrtp GOTO   =OUTRP          Output ")".
2468            *
2469 056D9 171   Outb,+ D1=D1+ 2
2470 056DC 31C2  Outby, LCASC  \,\
2471 056E0 6000  Outbyt GOTO   =outbyt
2472            *
2473 056E4 3172  'DC     LCASC  \'\
2474 056E8 6000         GOTO   =outbyt
2475            *
2476 056EC 31A3  :DC     LCASC  \:\
2477 056F0 6000         GOTO   =outbyt
2478            *
2479            ********************************************************************
2480            ********************************************************************
2481            **
2482            ** Name:    DC=EXP  -  Decompile '=' & Expression
2483            **
2484            ** Category:   DCMUTL
2485            **
2486            ** Purpose:
2487            **      Outputs ascii =, then decompiles expression
2488            **
2489            ** Entry:
2490            **      P       =  0
2491            **      D1 2 nibbles prior to expression
2492            **      D0 output pointer
2493            **      D(A) contains available memory end (AVMEME)
```

```
2494            **
2495            ** Exit:
2496            **      P       =  0
2497            **      D1 past expression
2498            **      via EXPRDC
2499            **
2500            ** Calls:      Outbyt
2501            **
2502            ** Uses.......
2503            **  Exclusive: A,C, D1,D0
2504            **  Inclusive: A-C, R0-R2, D1,D0, S0,S3,S8,S10,S11 - EXPRDC
2505            **
2506            ** Stk lvls:   4
2507            **
2508            ** History:
2509            **
2510            **    Date      Programmer              Modification
2511            **  --------    ----------   ----------------------------------
2512            **  08/24/82    S.W.         Created routine
2513            **
2514            ********************************************************************
2515            ********************************************************************
2516            *
2517 056F4 31D3  DC=EXP LCASC  \=\
2518 056F8 171   OB+EXD D1=D1+ 2              step over token
2519 056FB 7000  OBEXPR GOSUB  =outbyt        Output char
2520 056FF 5E1          GONC   Exprdc         (B.E.T.)
2521            *
```

```
2522                    STITLE LBLDC - Label Reference Decompile
2523                ************************************************************
2524                ************************************************************
2525                **
2526                ** Name:(S) LABLDC  -  Lable Decompile
2527                **
2528                ** Category:   DCMUTL
2529                **
2530                ** Purpose:
2531                **      Decompiles label references
2532                **
2533                ** Entry:
2534                **      D1 at tLBLRF
2535                **      D0 output pointer
2536                **      P=0
2537                **      D(A) contains available memory end (AVMEME)
2538                **
2539                ** Exit:
2540                **      P=0
2541                **      Carry clear
2542                **       D1 past string expression or literal
2543                **       D0 past decompiled label
2544                **
2545                **      If string expression, through EXPRDC
2546                **                           else through OUTBYT
2547                **
2548                ** Calls:      AD1+2, ASCICK, OUTBYT
2549                **
2550                ** Uses.....
2551                **  Inclusive: A,C, D1,D0
2552                **  Exclusive: A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 - EXPRDC
2553                **
2554                ** Stk lvls:   4
2555                **
2556                ** Detail:
2557                **      tLBLRF tLITRL <asci label>
2558                **      tLBLRF <string expression>
2559                **
2560                ** History:
2561                **
2562                **    Date     Programmer             Modification
2563                **  --------   ----------   ---------------------------------
2564                **  07/13/82   J.P.         Modified documentation
2565                **
2566                ************************************************************
2567                ************************************************************
2568 05702 747F =LABLDC GOSUB  AD1+2           Label Token
2569 05706 3100        LC(2)  =tLITRL         Label Literal ?
2570 0570A 966         ?A#C   B
2571 0570D 11          GOYES  Exprdc
2572 0570F 71DF        GOSUB  'DC
2573 05713 743A        GOSUB  ASCI+
2574 05717 5CC         GONC   'DC             (B.E.T.)
2575                *
2576                * Label is a string expression
```

```
2577                *
2578 0571A 747A  Exprd* GOSUB  GTXT++        Want leading & trailing blanks
2579 0571E 6000  Exprdc GOTO   =EXPRDC       Decompile String expression
```

```
2580                    STITLE Decompile File commands
2581                ********************************************************************
2582                ********************************************************************
2583                **
2584                ** Name:    RENMDC  -  PURGE, COPY Decompile
2585                ** Name:(S) PURGDC  -  PURGE, COPY Decompile
2586                ** Name:    COPYDC  -  PURGE, COPY Decompile
2587                **
2588                ** Category: STDCMP
2589                **
2590                ** Purpose:  Decompiles RENAME, PURGE, PRIVATE, COPY
2591                **
2592                ** Entry:    P= 0
2593                **           D1 past begin BASIC token
2594                **           D0 output file
2595                **           D(A) contains available memory end (AVMEME)
2596                **
2597                ** Exit:     via OUTEL1
2598                **
2599                ** Calls:    FILDC, GTEXT+, EOLXC*, BLNKCK
2600                **
2601                ** Uses:     A-C, D1,D0, R1,R2, S8,S9
2602                **           A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
2603                **
2604                ** Stk lvls: 6
2605                **
2606                ** Detail:   NAME <file name>
2607                **           PURGE <file specifier>|ALL|keys
2608                **           RENAME [<file specifier>|keys] TO <file name>|keys
2609                **           COPY [ <file spec> | KEYS | CARD | PCRD ] [ TO
2610                **                [ <file spec> | KEYS | CARD | PCRD ] ]
2611                **
2612                ** Note:     The TO clause is OPTIONAL in a COPY statement
2613                **             In RENAME, the TO is ALWAYS there
2614                **
2615                **           <destination file> is optional in COPY
2616                **          This requires an EOL Check to be done after <file2>
2617                **             This does not affect RENAME Decompile
2618                **
2619                ** History:
2620                **
2621                **    Date     Programmer   Modifications
2622                **  --------   ----------   -------------
2623                **  08/18/82   S.W.         Added documentation
2624                **
2625                ********************************************************************
2626                ********************************************************************
2627                *
2628 05722 76CB  =COPYDC GOSUB  EOLXC*
2629 05726 3100  =RENMDC LC(2)  =tTO           TO TOKEN
2630 0572A 962           ?A=C   B
2631 0572D E0            GOYES  RNMDC2
2632                *
2633                * Decompile <file spec1> | Reserve Word
2634                *
```

```
2635 0572F 7620          GOSUB  FILDC*
2636 05733 7A8A          GOSUB  BLNKCK         OUTPUT BLANK AFTER S.VAR OR LABEL
2637              *
2638              *   TO is optional in COPY
2639              *
2640 05737 71BB          GOSUB  EOLXC*
2641 0573B 859    RNMDC2 ST=1   9              WANT TRAILING BLANK
2642              *
2643 0573E 736A   RNMDC3 GOSUB  GTEXT1
2644 05742 5B0           GONC   PRGDC1         (B.E.T.)
2645              *
2646 05745 3100 =PURGDC LC(2)  =tALL           ALL TOKEN
2647 05749 962           ?A=C   B
2648 0574C 2F            GOYES  RNMDC3         NO BLANKS
2649              *
2650              * Check if EOL for:
2651              *   PURGE
2652              *   COPY <file1> TO
2653              *
2654 0574E 7A9B   PRGDC1 GOSUB  EOLXC*         No return if at end of statement
2655 05752 629D          GOTO   NAMEDC         Call FILDC*, OUTEL1
2656              *
```

```
2657                    STITLE Decompile Filename/Device id
2658                *
2659                ****************************************************************
2660                ****************************************************************
2661                **
2662                ** Name:(S) FILDC*  -  File Decompile
2663                **
2664                ** Category:   DCMUTL
2665                **
2666                ** Purpose:    Decompiles mainframe file specifiers & HPIL file
2667                **             specifiers if HPIL plugged in
2668                **
2669                ** Entry:
2670                **                  P=0
2671                **                  D(A) contains available memory end
2672                **                  D0   output pointer
2673                **                  2 entry points:
2674                **                  1) FILDC+ - D1 hasn't yet been incremented.
2675                **                  2) FILDC* - D1 already at file spec
2676                **
2677                ** Exit:       D1 past file specifier
2678                **             File specifier decompiled, with D0 updated
2679                **             P=0
2680                **
2681                ** Calls:    POLLD+, OUTNBS, ASCICK, EXPRDC, OUT1TK, GTEXT+,
2682                **           FINDA, D=AVME
2683                **
2684                ** Uses:     S8,S9, A-C, D1,D0, R1,R2
2685                **           A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
2686                **
2687                ** Stack lvls: 5
2688                **
2689                ** Detail:   Will check for tKEYS, tCARD, tPCRD
2690                **
2691                **           Assumes that non-mainframe file specs are
2692                **           tokenized with preceding tCOLON.
2693                **
2694                **           Must immediately precede SKIPDC code, since it
2695                **           falls into SKIPDC.
2696                **
2697                ** History:
2698                **
2699                **    Date     Programmer   Modifications
2700                ** --------   ----------   -------------
2701                **  07/07/82   S.W.         Improved documentation
2702                **
2703                ****************************************************************
2704                ****************************************************************
2705                *
2706                ****************************************************************
2707                ****************************************************************
2708                **
2709                ** Name:(S) pFILDC  -  Polls for File Decompile
2710                **
2711                ** Category:   POLL
```

```
2712            **
2713            ** Type:       POLL
2714            **
2715            ** Purpose:
2716            **      Polls for handler for device decompile
2717            **
2718            ** Should poll be "Handled" (return with XM=0)?:
2719            **      Yes
2720            **
2721            ** Meaning of "Handling" Poll (what does code do if handled?):
2722            **      Decompiled device specifier output & D0 updated.
2723            **
2724            ** Entry conditions for handler (registers, ST, RAM, etc.):
2725            **      Carry clear
2726            **      B[A] = Poll number.
2727            **      HEX mode.
2728            **      P=0.
2729            **      D1 at tCOLON
2730            **      A(B) contains tCOLON
2731            **      D0 past last decompiled character
2732            **      D(A) contains the end of available memory
2733            **
2734            ** Normal exit conditions from handler if handled (ST, RAM,
2735            ** registers, etc.):
2736            **      P=0
2737            **      Carry clear
2738            **      HEX mode.
2739            **      XM=0.
2740            **      D1 past the file/device specifier
2741            **      File specifier output & D0 updated
2742            **      D(A) preserved
2743            **
2744            ** Normal exit conditions from handler if not handled (ST, RAM,
2745            ** registers, etc.):
2746            **      P=0
2747            **      Carry clear
2748            **      HEX mode.
2749            **      XM=1.
2750            **
2751            ** Error exit conditions from handler:
2752            **  (Only happens with insufficient memory)
2753            **      P=0
2754            **      Carry set.
2755            **      HEX mode.
2756            **
2757            ** Available subroutine levels:
2758            **      6
2759            **
2760            ** NOTE:
2761            **      When D(A) is passed to the poll handler, it reflects what
2762            **       the end of available memory WILL be once we get to the
2763            **       handler.
2764            **
2765            ** What registers/RAM may be used if handled?:
2766            **      A-D, D0, D1, P always available
```

```
2767            **      S8,S9
2768            **      +Anything EXPRDC uses (R0,R1,R2,S0,S3,S10,S11)
2769            **
2770            ** What registers/RAM may be used if not handled?:
2771            **      A-D, D0, D1, P
2772            **      Same as if handled (see above)
2773            **
2774            ** What registers/RAM may be used if error exit:
2775            **      A-D, D0, D1, P
2776            **      Same as if handled (see above)
2777            **
2778            ** Special memory/pointer considerations (are pointers funny?):
2779            **      No
2780            **
2781            ** Envisioned application(s):
2782            **      Decompile non-mainframe device
2783            **
2784            ** History:
2785            **
2786            **    Date     Programmer          Modification
2787            **  --------   ----------   ----------------------------------
2788            **  07/08/82   S.W.         Added documentation
2789            **
2790            ************************************************************************
2791            ************************************************************************
2792            *
2793 05756 171   FILDC+ D1=D1+ 2
2794 05759 848  =FILDC* ST=0   8
2795 0575C 7001  FLDC03 GOSUB  finda+          Read 6 nibs into A first
2796 05760 00           CON(2) =tLITRL
2797 05762 C20          REL(3) FLDC07
2798 05765 00           CON(2) =tKEYS
2799 05767 540          REL(3) FLDC25
2800 0576A 00           CON(2) =tCARD
2801 0576C C30          REL(3) FLDC20
2802 0576F 00           CON(2) =tPORT
2803 05771 730          REL(3) FLDC20
2804 05774 00           CON(2) =tMAIN
2805 05776 230          REL(3) FLDC20
2806 05779 FE           CON(2) =tPCRD
2807 0577B D10          REL(3) FLDC15
2808 0577E 00           CON(2) =tCOLON         HPIL
2809 05780 340          REL(3) FLDC70
2810 05783 00           CON(2) 00
2811            *
2812 05785 878          ?ST=1  8               Already parsed literal file name?
2813 05788 00           RTNYES
2814 0578A 6000         GOTO   =EXPRDC         Must be an expression
2815            *
2816 0578E 79B9  FLDC07 GOSUB  ASCI+
2817            *
2818 05792 858          ST=1   8               DON'T DO EXPRDC!
2819 05795 56C          GONC   FLDC03          (B.E.T.)
2820            *
2821            * XWORD token found
```

```
2822 05798 AF6   FLDC15 C=A    W
2823 0579B 35FE         LC(6)  =tPCRD
           10E3
2824 057A3 976          ?A#C   W
2825 057A6 00           RTNYES                  Not PCRD?
2826 057A8 704F  FLDC20 GOSUB  :DC
2827 057AC 849   FLDC25 ST=0   9               NO LEADING OR TRAILING BLANKS
2828 057AF 76AD         GOSUB  gtexta
2829             * CHECK FOR PORT#
2830 057B3 3182         LCASC  \(\
2831 057B7 966          ?A#C   B
2832 057BA 00           RTNYES
2833             * DECOMPILE PORT #
2834 057BC 171          D1=D1+ 2
2835 057BF 611F         GOTO   ?EXPR-
2836             *
2837             * Must set AVMEMS @ Current DO before Poll
2838             *   Prevents Poll Save Area from overwriting Decompiled Output
2839             *
2840 057C3 8E00  FLDC70 GOSUBL =polld+          Set AVMEMS @ Current DO
           00
2841             *                              & Update D(A) during poll
2842 057C9 20           CON(2) =pFILDC
2843 057CB 445          GOC    FLDCER           HANDLED & ERRORED?
2844 057CE 8F00         GOSBVL =D=AVME          Reset D(A) (AVMEME)
           000
2845 057D5 831          ?XM=0                   HANDLED & OKAY?
2846 057D8 00           RTNYES
2847             * DEFAULT HANDLER FOR DECOMPILING UNRECOGNIZED FILE SPEC
2848 057DA 7E0F         GOSUB  :DC
2849 057DE 3F56         LCASC  \lanretxe\
           8747
           5627
           E616
           C6
2850 057F0 7D2C         GOSUB  OUTC15
2851 057F4 07           C=RSTK                  Pop level off stack
2852             *
2853             *       Falls into SKIPDC
2854             *   !!!!!!!!!!!!!!!!!!!!!!!!!!!!
2855             *
2856             * Unrecognized XWORD
2857             *    Skip to End of Statement
2858             *    Go Decompile End of Line Terminator
2859             *
2860             ****************************************************************
2861             ****************************************************************
2862             **
2863             ** Name:(S) SKIPDC  -  Skip Rest of Statement Decompile
2864             **
2865             ** Category:   DCMUTL
2866             **
2867             ** Purpose:
2868             **       When an unrecognized token is encountered, decompile
2869             **       of that statement cannot continue.  SKIPDC skips
```

```
2870                **      D1 to the end of that statement.
2871                **
2872                ** Entry:
2873                **      (INADDR) = Address of the statement length byte of
2874                **                 the statement currently being decompiled.
2875                **
2876                ** Exit:
2877                **      D1 points to the statement terminator byte in the
2878                **      token stream.
2879                **      Exit is via OUTEL1.
2880                **      A(A)= Statement Length for the statement skipped.
2881                **      C(A)= D1
2882                **
2883                ** Calls:      None
2884                **
2885                ** Uses:       A(A), C(A), D1
2886                **
2887                ** Stk lvls:   0
2888                **
2889                ** Detail:     Must immediately follow FILDC
2890                **
2891                ** History:
2892                **
2893                **    Date     Programmer              Modification
2894                **  --------   ----------   ----------------------------------
2895                **  11/08/83   S.W.         Added documentation header
2896                **
2897                ********************************************************************
2898                ********************************************************************
2899                *
2900 057F6 136     =SKIPDC CDOEX                   SAVE D0
2901 057F9 1B00            D0=(5) =INADDR
           000
2902 05800 142             A=DAT0 A                CURRENT STMT LENGTH ADDRESS
2903 05803 130             D0=A
2904 05806 D0              A=0    A
2905 05808 14A             A=DAT0 B                STATEMENT LENGTH
2906 0580B 136             CDOEX
2907 0580E C2              C=C+A  A
2908 05810 135             D1=C                    POINT TO @ OR EOL
2909 05813 6CEA            GOTO   OUTEL1
2910                *
2911
2912                *-----------------------------------
2913 05817 181     out2t0  D0=D0- 2
2914 0581A 8C00    =out2tc GOLONG =OUT2TC
           00
2915                *-----------------------------------
2916
2917                *
2918 05820 8C00    FLDCER  GOLONG =MFERR
           00
2919                *
```

```
2920                    STITLE LIST Decompile
2921                ********************************************************************
2922                ********************************************************************
2923                **
2924                ** Name:(S) LISTDC  -  Decompiles LIST, RENUMBER, SECURE, MERGE
2925                **
2926                ** Category:   DCMUTL
2927                **
2928                ** Purpose:  DECOMPILES LIST, SECURE, MERGE STATEMENTS
2929                **
2930                ** Entry:    P= 0
2931                **           D1 past begin BASIC token
2932                **           D0 output pointer
2933                **           D(A) contains available memory end (AVMEME)
2934                **
2935                ** Exit:     via OUTELA
2936                **
2937                ** Calls:    FILDC, LIN#DC, EOLXC*, COMTST, OUTBYT
2938                **
2939                ** Uses:     A-C, D1,D0, S8,S9, R1,R2
2940                **           A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
2941                **
2942                **
2943                ** Detail:   EXPECTS THAT S8 WILL BE CLEAR UPON ENTRY
2944                **
2945                ** History:
2946                **
2947                **    Date     Programmer          Modifications
2948                **  --------   -----------  ------------------------
2949                **  08/29/83   S.W.         Added documentation header
2950                **
2951                ********************************************************************
2952                ********************************************************************
2953                *
2954                *
2955 05826 868   LSTDC+ ?ST=0  8              NO COMMA NECESSARY?
2956 05829 60           GOYES  LSTDC1
2957                *
2958 0582B 7DAE         GOSUB  Outby,
2959 0582F 171   LSTDC1 D1=D1+ 2              STEP OVER tCOMMA
2960 05832 7FD8         GOSUB  LIN#DC
2961                *
2962 05836 858   LSTDC* ST=1   8              NEXT TIME THRU COMMA NEEDED
2963                *
2964 05839       =RNUMDC
2965 05839       =SECRDC
2966 05839 7FAA =LISTDC GOSUB  EOLXC*         No return if at stmt end
2967 0583D 772A         GOSUB  COM10
2968 05841 44E          GOC    LSTDC+
2969                *
2970 05844 711F =MERGDC GOSUB  FILDC*
2971 05848 6DEF         GOTO   LSTDC*
2972                *
2973                *
2974                *
```

```
2975                    STITLE Decompile USER | LC stmt
2976                ********************************************************************
2977                ********************************************************************
2978                **
2979                ** Name:     USERDC  -  USER and Lower Case Decompile
2980                ** Name:     LCDC    -  USER and Lower Case Decompile
2981                **
2982                ** Category:   STDCMP
2983                **
2984                ** Purpose:  Decompile USER | LC statement
2985                **
2986                ** Entry:    D1 past tUSER or tFLIP
2987                **           D0 output pointer
2988                **           D(A) contains available memory end (AVMEME)
2989                **           A(B) contains token pointed to by D1
2990                **           S8=0
2991                **
2992                ** Exit:     Through OUTELA
2993                **
2994                ** Calls:    GTEXT1, EXPRDC, FINDA
2995                **
2996                ** Uses:     A-C, D1,D0, S9, R1,R2
2997                **           A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
2998                **
2999                ** Stk lvls: 5
3000                **
3001                ** Detail:   USER [ ( ON | OFF ) ]
3002                **           LC [ ( ON | OFF ) ]
3003                **
3004                ** NOTE:     Decompile for LC is called FLIPDC because the
3005                **           statement used to be called FLIP
3006                **
3007                ** History:
3008                **
3009                **    Date     Programmer   Modifications
3010                **  --------   ----------   -------------
3011                **  08/18/82   S.W.         Added documentation
3012                **
3013                ********************************************************************
3014                ********************************************************************
3015                *
```

```
3016                    STITLE BEEP Decompile
3017            ********************************************************************
3018            ********************************************************************
3019            **
3020            ** Name:    BEEPDC  -  BEEP Statement Decompile
3021            **
3022            ** Category:   STDCMP
3023            **
3024            ** Purpose:
3025            **      Decompile BEEP statement
3026            **
3027            ** Entry:
3028            **      D1 past BEEP token
3029            **      D0 output pointer
3030            **      D(A) contains available memory end (AVMEME)
3031            **      A(B) contains token pointed to by D1
3032            **      S8=0
3033            **
3034            ** Exit:
3035            **      Through OUTELA
3036            **      P=0
3037            **
3038            **      Through USRDC1 if BEEP ON | OFF
3039            **
3040            ** Calls:       EXPRDC, FINDA, GTEXT1
3041            **
3042            ** Uses.......
3043            **  Exclusive: A-C, D1,D0, R1,R2, S9
3044            **  Inclusive: A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
3045            **
3046            ** Stl lvls:   5
3047            **
3048            ** Detail:
3049            **      BEEP [ ON | OFF ]
3050            **      BEEP [ <frequency> [ , <duration> ]
3051            **
3052            ** Algorithm:
3053            **
3054            **      If next char = End of Line terminator  A(1)#F
3055            **         go Output End of line terminator (OUTELA)
3056            **      Set BEEP entry flag for USERDC
3057            **      If next token = ON | OFF            (goto USRDC1)
3058            **         go Output "ON" | "OFF" and EOL
3059            **      else
3060            **         Position back to expression
3061            **         Decompile frequency & possible duration (EXPRDC)
3062            **         go Output End of line terminator
3063            **
3064            ** History:
3065            **
3066            **    Date     Programmer              Modification
3067            **  --------   ----------   ------------------------------------
3068            **  07/13/82   J.P.         Modified documentation
3069            **
3070            ********************************************************************
```

```
3071            ********************************************************************************
3072            *
3073 0584C      =BEEPDC
3074 0584C      =FLIPDC
3075            *
3076            * BEEP ON | OFF entry
3077            *
3078 0584C      =USERDC
3079 0584C 7410  USRDC1 GOSUB  finda
3080 05850 00           CON(2) =tON
3081 05852 AAA          REL(3) TRACDC
3082 05855 00           CON(2) =tOFF
3083 05857 5AA          REL(3) TRACDC
3084 0585A 00           CON(2) 0
3085            *
3086 0585C 631C         GOTO   DROPDC         Optional expression(s)
3087            *
3088 05860 15B5  finda+ A=DAT1 6
3089 05864 8C00 =finda  GOLONG =FINDA
           00
3090            *
3091            *
```

```
3092                    STITLE SUB Decompile
3093                ****************************************************************
3094                ****************************************************************
3095                **
3096                ** Name:     SUBDC   -  SUB statement decompile
3097                **
3098                ** Category: STDCMP
3099                **
3100                ** Purpose:  Decompiles SUB statement
3101                **
3102                ** Entry:    P= 0
3103                **           D(A) contains available memory end (AVMEME)
3104                **           D1 past tSUB
3105                **           D0 output pointer
3106                **
3107                ** Exit:     via OUTEL1
3108                **
3109                ** Calls:    PARMCK, EXPRDC, OUT3TK, OUTBYT, REMP (Rem parse)
3110                **           AD1+2, COMTST, !TEST
3111                **
3112                ** Uses:     A-C, D1,D0, S8,S9
3113                **           A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
3114                **
3115                ** Stk lvls: 5
3116                **
3117                ** Detail:   MUST HAVE S8 CLEAR TO DECOMPILE DUMMY ARRAYS, AS
3118                **           PER CONSTRAINTS OF EXPDDA
3119                **
3120                ** History:
3121                **
3122                **    Date     Programmer   Modifications
3123                **  --------   ----------   -------------
3124                **  08/18/82   S.W.         Added documentation
3125                **  04/29/83   S.W.         Fixed bug with ! after SUB
3126                **                           by calling REMP (REM parse)
3127                **                           instead of ASCICK
3128                **
3129                ****************************************************************
3130                ****************************************************************
3131                *
3132                *
3133                *
3134 0586A 174   =SUBDC  D1=D1+ 5              STEP OVER 5 NIB FIELD
3135 0586D 79FD          GOSUB  PARMCK
3136 05871 502    SUBDC1 GONC   SUBDC6         DONE WITH STATEMENT?
3137                * MUST BE VARIABLE
3138 05874 7000          GOSUB  =EXPRDC       MAY BE DUMMY ARRAY
3139 05878 7CE9          GOSUB  COM10
3140 0587C 5E0           GONC   SUBDC5
3141                *
3142 0587F 77FD          GOSUB  AD1+2         Step over comma; read next tok
3143 05883 771E          GOSUB  PRMCK4
3144 05887 69EF          GOTO   SUBDC1
3145                *
3146 0588B 764E   SUBDC5 GOSUB  Outrtp
```

```
3147 0588F 171          D1=D1+ 2            Step over tPRMEN
3148 05892 14B   SUBDC6 A=DAT1 B
3149 05895 7D7B         GOSUB  !TEST
3150 05899 5C1          GONC   SUBDC7
3151            *
3152 0589C 3502         LCASC  \ ! \
           1202
3153 058A4 8E00         GOSUBL =OUT3TC
           00
3154 058AA 8E00         GOSUBL =REMPO5
           00
3155 058B0 181          D0=D0- 2            Don't want OD output
3156 058B3 173          D1=D1+ 4            Step off OD & dummy tEOL
3157            *
3158 058B6 174   SUBDC7 D1=D1+ 5            STEP OVER 5 NIB FIELD
3159 058B9 543          GONC   CALLD8       (B.E.T.)
```

```
3160                    STITLE CALL Decompile
3161                ********************************************************************
3162                ********************************************************************
3163                **
3164                ** Name:    CALLDC  -  CALL Decompile
3165                **
3166                ** Category: STDCMP
3167                **
3168                ** Purpose:  Decompiles CALL statement
3169                **
3170                ** Entry:    D1 past tCALL
3171                **           D0 output pointer
3172                **           D(A) contains available memory end (AVMEME)
3173                **
3174                ** Exit:     via OUTEL1
3175                **
3176                ** Calls:    PARMCK, EXPRDC, OUTNBC, FILDC
3177                **
3178                ** Uses:     A-C, D1,D0, S8,S9
3179                **           A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
3180                **
3181                ** Stk lvls: 5
3182                **
3183                ** History:
3184                **
3185                **    Date     Programmer   Modifications
3186                **  --------   ----------   -------------
3187                **  08/18/82   S.W.         Added documentation
3188                **
3189                ************************************************
3190                ************************************************
3191                *
3192 058BC 77AD  =CALLDC GOSUB  PRMCK+
3193 058C0 5D0           GONC   CALLD2
3194                * MUST BE AN EXPRESSION
3195 058C3 7000   CALLD1 GOSUB  =EXPRDC        MAY BE A DUMMY ARRAY
3196                * Equivalent to call to PRMCK3
3197                * Assume S9 is still set from PRMCK+ call above
3198                * Saves D1=D1+ 2 instruction
3199 058C7 76ED          GOSUB  PRMCK7         Read in token,etc
3200 058CB 47F           GOC    CALLD1
3201 058CE 14B   CALLD2  A=DAT1 B              Get token
3202 058D1 3100          LC(2)  =tIN
3203 058D5 966           ?A#C   B
3204 058D8 61            GOYES  CALLD8
3205                * Output leading & trailing blanks
3206 058DA 3702          LCASC  \ NI \
           94E4
           02
3207 058E4 27            P=     7
3208 058E6 793B          GOSUB  OUTNBC
3209 058EA 786E          GOSUB  FILDC+
3210 058EE 611A  CALLD8  GOTO   OUTEL1
3211                *
3212                *
```

```
3213                    STITLE STAT Decompile
3214                ****************************************************************
3215                ****************************************************************
3216                **
3217                ** Name:    STATDC  -  STAT Decompile
3218                **
3219                ** Category: STDCMP
3220                **
3221                ** Purpose:  Decompiles STAT statement
3222                **
3223                ** Entry:    P= 0
3224                **           D1 past tSTAT
3225                **           D0 output pointer
3226                **           D(A) contains available memory end (AVMEME)
3227                **
3228                ** Exit:     via OUTEL1
3229                **
3230                ** Calls:    -EXPR-, VARDC
3231                **
3232                ** Uses:     A-C, D1,D0, S6
3233                **           A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
3234                **
3235                ** Stk lvls: 6
3236                **
3237                ** History:
3238                **
3239                **    Date     Programmer   Modifications
3240                ** --------   ----------   -------------
3241                ** 08/18/82   S.W.         Added documentation
3242                **
3243                ****************************************************
3244                ****************************************************
3245                *
3246                *
3247                *
3248 058F2 768A =STATDC GOSUB  VARDC
3249 058F6 866         ?ST=0  6              NO ARRAY TOKEN?
3250 058F9 41          GOYES  UNDC10         Could go to OUTELA
3251 058FB 7ECD        GOSUB  -EXPR-
3252 058FF 5D0         GONC   UNDC10         (B.E.T.)
```

```
3253                    STITLE UNPROTECT Decompile
3254                ********************************************************************
3255                ********************************************************************
3256                **
3257                ** Name:    UNPRDC  -  UNPROTECT Decompile
3258                **
3259                ** Category: STDCMP
3260                **
3261                ** Purpose:  Decompile UNPROTEC(T)
3262                **
3263                ** Entry:    UNPROTEC has been decompiled.
3264                **           D1 past the begin BASIC token
3265                **           D0 output pointer
3266                **           D(A) contains available memory end (AVMEME)
3267                **           P= 0
3268                **
3269                ** Exit:     via OUTEL1
3270                **
3271                ** Calls:    OUTBYT
3272                **
3273                ** Uses:     A,C, D0,D1
3274                **
3275                ** Stk lvls: 2
3276                **
3277                ** History:
3278                **
3279                **    Date     Programmer        Modifications
3280                **  --------   ----------   ------------------------
3281                **  08/29/83   S.W.         Updated documentation
3282                **
3283                ********************************************************************
3284                ********************************************************************
3285                *
3286 05902 181  =UNPRDC D0=D0- 2              ELIMINATE BLANK
3287 05905 3145         LCASC  \T\
3288 05909 7000         GOSUB  =outbyt       OUTPUT "T"
3289 0590D 62F9  UNDC10 GOTO   OUTEL1
3290                ********************************************************************
3291                ********************************************************************
3292                **
3293                ** Name:    RESETd  -  RESET decompile
3294                **
3295                ** Category:   STDCMP
3296                **
3297                ** Purpose:
3298                **      Decompile RESET statement.
3299                **
3300                ** Entry:
3301                **      D1 past RESET token.
3302                **      D0 output pointer
3303                **      D(A) contains available memory end (AVMEME)
3304                **      A(5-0) contains token pointed to by D1
3305                **      A(15-6) = C(15-6)
3306                **      P= 0
3307                **
```

```
3308            ** Exit:
3309            **      Through OUTELA.
3310            **
3311            ** Calls:      GTEXT1
3312            **
3313            ** Uses.......
3314            **  Exclusive: A-C, D1,D0, R1,R2, S9
3315            **  Inclusive: A-C, D1,D0, R0-R2, S0,S3,S8,S10,S11 -- EXPRDC
3316            **
3317            ** Stk lvls:   4
3318            **
3319            ** History:
3320            **
3321            **    Date      Programmer              Modification
3322            **  --------    ----------   ----------------------------------
3323            **  09/21/82    NM           Wrote
3324            **
3325            ************************************************************************
3326            ************************************************************************
3327 05911 35FE =RESETd LC(6)  =tCLOCK
           1051
3328 05919 976          ?A#C   W              Find CLOCK token?
3329 0591C 1F           GOYES  UNDC10
3330 0591E 6DD9         GOTO   TRACDC         Output CLOCK token
```

```
 !DC      Abs   21317 #05345 -  1338   237  1199
 !TEST    Abs   21526 #05416 -  1534   235  1334  3149
 'DC      Abs   22244 #056E4 -  2473   206   208  2572  2574
 -EXPR-   Abs   22221 #056CD -  2465  1711  3251
 :DC      Abs   22252 #056EC -  2476   209  2826  2848
 ?EXPR-   Abs   22225 #056D1 -  2466  2835
 AD1+2    Abs   22138 #0567A -  2385   186   258   635  2568  3142
 ADCH++   Abs   20787 #05133 -   634  1448
 ADDCH+   Abs   20790 #05136 -   635  1434  1447
 ADDCHR   Abs   20794 #0513A -   636  1452
=ADDDC    Abs   21616 #05470 -  1676
 ARANGE   Ext               -  1441
=ARYDC    Abs   20856 #05178 -   756  1101
 ARYDC+   Abs   20837 #05165 -   748   968  1103
 ARYDC2   Abs   20873 #05189 -   761   759
=ASCO2    Abs   20826 #0515A -   692
 ASCO3    Abs   20833 #05161 -   695   690
 ASCI+    Abs   20811 #0514B -   686   693  2573  2816
=ASCICK   Abs   20814 #0514E -   687   207
 ASNDO6   Abs   21737 #054E9 -  1778  1926
=ASSNDC   Abs   21705 #054C9 -  1763
=BEEPDC   Abs   22604 #0584C -  3073
 BLNKC1   Abs   20933 #051C5 -   859   862
=BLNKCK   Abs   20929 #051C1 -   857  1306  1337  2276  2323  2636
 CALLD1   Abs   22723 #058C3 -  3195  3200
 CALLD2   Abs   22734 #058CE -  3201  3193
 CALLD8   Abs   22766 #058EE -  3210  3159  3204
=CALLDC   Abs   22716 #058BC -  3192
 COM1O    Abs   21096 #05268 -   991  1678  2414  2967  3139
 COMCK1   Ext               -   991
 COMTST   Abs   21093 #05265 -   990  1922
=COPYDC   Abs   22306 #05722 -  2628
=CREADC   Abs   22094 #0564E -  2321
=DO=OBS   Abs   20583 #05067 -   337
 DOASCI   Ext               -   590
=DOOUTB   Abs   20583 #05067 -   338
 D1C=R3   Ext               -   918
 D=AVME   Ext               -   336  2844
=DATADC   Abs   21330 #05352 -  1342
 DC=EXP   Abs   22260 #056F4 -  2517  1048  1143
 DCDONE   Abs   21361 #05371 -  1358  1354
=DECDC    Abs   21127 #05287 -  1096  1108
 DECDC2   Abs   21147 #0529B -  1103  1099
=DEFADC   Abs   21244 #052FC -  1297
=DEFDC    Abs   21014 #05216 -   964
 DEFDC1   Abs   21041 #05231 -   971   969
 DEFDC2   Abs   21060 #05244 -   977   983
 DEFDC4   Abs   21080 #05258 -   985   980
 DEFDC6   Abs   21086 #0525E -   986   975
=DELAYd   Abs   21651 #05493 -  1699
=DISPDC   Abs   21584 #05450 -  1662
=DROPDC   Abs   21616 #05470 -  1675  1490  1664  1669  1701  1713  1948  2328
                              3086
=DSTRDC   Abs   21120 #05280 -  1094
 ELSEDC   Abs   21218 #052E2 -  1208  1318
```

```
=ENDDC   Abs   21240 #052F8 -  1295
=ENDDDC  Abs   21960 #055C8 -  2167
 ENDS10  Abs   21949 #055BD -  2163  2170
 ENDS11  Abs   21956 #055C4 -  2165  2223  2225
=ENDSDC  Abs   21928 #055A8 -  2160
=EOLDC   Abs   21506 #05402 -  1527  1290
=EOLXC*  Abs   21228 #052EC -  1290   986  1105  1147  1295  1676  2213  2325
                               2628  2640  2654  2966
=EOLXCK  Abs   21509 #05405 -  1528
 ETMR10  Abs   21858 #05562 -  2025  2014
 ETMR20  Abs   21875 #05573 -  2034  1763
=ETMRDC  Abs   21837 #0554D -  2011  1897  2077
 EXPRDC  Ext                  1047  1190  1700  1905  2212  2215  2579  2814
                               3138  3195
 Exprd*  Abs   22298 #0571A -  2578  1146  1150
 Exprdc  Abs   22302 #0571E -  2579  2520  2571
=FILDC*  Abs   22361 #05759 -  2794  1777  2275  2327  2379  2635  2970
 FILDC+  Abs   22358 #05756 -  2793  1919  3209
 FINDA   Ext                  3089
=FIXDC   Abs   21651 #05493 -  1700  1657  1659  1720  2129
 FLDC03  Abs   22364 #0575C -  2795  2819
 FLDC07  Abs   22414 #0578E -  2816  2797
 FLDC15  Abs   22424 #05798 -  2822  2807
 FLDC20  Abs   22440 #057A8 -  2826  2801  2803  2805
 FLDC25  Abs   22444 #057AC -  2827  2799
 FLDC70  Abs   22467 #057C3 -  2840  2809
 FLDCER  Abs   22560 #05820 -  2918  2843
=FLIPDC  Abs   22604 #0584C -  3074
=FNCK    Abs   20956 #051DC -   909  1045
=FNDC    Abs   21017 #05219 -   965  1046
=FORDC   Abs   21162 #052AA -  1142
 FTYPD+  Ext                  2321
=GOSBDC  Abs   21806 #0552E -  1932
=GOTODC  Abs   21806 #0552E -  1931
=GTEXT   Abs   20601 #05079 -   437
=GTEXT+  Abs   20889 #05199 -   842  1095  2019  2278
=GTEXT1  Abs   20901 #051A5 -   847  1209  1297  1710  2032  2123  2643
 GTEXTI  Abs   20598 #05076 -   436   241   847
=GTEXTM  Abs   20619 #0508B -   443
=GTEXTX  Abs   20635 #0509B -   454   439   442
=GTXT++  Abs   20882 #05192 -   839  1192  1767  1907  2578
 GTXT+1  Abs   20929 #051C1 -   858   182   273   840   854
 GTXT+3  Abs   20951 #051D7 -   867   848
 GTXT10  Abs   20650 #050AA -   465   446
=IFDC    Abs   21185 #052C1 -  1190
 IFDC1   Abs   21193 #052C9 -  1193  1210
 INADDR  Ext                   178  2901
=INPTDC  Abs   21573 #05445 -  1656
=KEYDC   Abs   21984 #055E0 -  2209
 KEYDC1  Abs   22042 #0561A -  2224  1775  2219
=LABLDC  Abs   22274 #05702 -  2568  1943
 LASTFN  Ext                   228
=LDCEXT  Abs   20318 #04F5E -   154
 LDCIF   Abs   20359 #04F87 -   177   222
=LDCM10  Abs   20335 #04F6F -   163   156
```

```
 LDCM20  Abs   20436 #04FD4 -   220   213  1206
 LDCM30  Abs   20446 #04FDE -   227   204
 LDCM50  Abs   20461 #04FED -   235   230
 LDCM60  Abs   20472 #04FF8 -   241   236
 LDCM80  Abs   20536 #05038 -   271   242
=LDCOMP  Abs   20329 #04F69 -   162
=LDCSET  Abs   20576 #05060 -   336   167
 LDCSPC  Ext                -   175
=LDSST1  Abs   20338 #04F72 -   167
=LDSST2  Abs   20382 #04F9E -   186
=LETDC   Abs   21102 #0526E - 1045   231
 LETDC3  Abs   21113 #05279 - 1048   988
=LIN#A+  Abs   20773 #05125 -   589
=LIN#AU  Abs   20770 #05122 -   588   263   266
 LIN#CK  Abs   20745 #05109 -   578  1668  1935
=LIN#D+  Abs   20754 #05112 -   582
=LIN#DC  Abs   20757 #05115 -   583   173  2960
=LISTDC  Abs   22585 #05839 - 2966
=LRDC    Abs   21651 #05493 - 1696
 LSTDC*  Abs   22582 #05836 - 2962  2971
 LSTDC+  Abs   22566 #05826 - 2955  2968
 LSTDC1  Abs   22575 #0582F - 2959  2956
 LSTLEN  Ext                - 1359
 MAINT   Ext                -  443
 MEMERR  Ext                - 1595
=MERGDC  Abs   22596 #05844 - 2970
 MFERR   Ext                - 2918
 MTADR+  Ext                -  484
=NAMEDC  Abs   21733 #054E5 - 1777  1773  2655
=NXTDC   Abs   21130 #0528A - 1097
 OB+EXD  Abs   22264 #056F8 - 2518  2406
 OBEXPR  Abs   22267 #056FB - 2519   756  2035  2466
=OFFDC   Abs   21885 #0557D - 2077
=ONDC    Abs   21741 #054ED - 1897
 ONDC10  Abs   21757 #054FD - 1905  1900
=ONDC20  Abs   21761 #05501 - 1907  1898
 ONDC40  Abs   21784 #05518 - 1922  1936  1944
 ONDC42  Abs   21799 #05527 - 1927  1923
 ONDC50  Abs   21809 #05531 - 1934  1925  1928
 ONDC80  Abs   21833 #05549 - 1948  1942
=OPTDC   Abs   21893 #05585 - 2118
 OPTDC1  Abs   21905 #05591 - 2122  2120
 OPTDC2  Abs   21924 #055A4 - 2129  2125
 OUT1T+  Ext                -  985
 OUT2TC  Ext                - 2914
 OUT3TC  Ext                - 3153
 OUTBS   Ext                -  338
 OUTBY+  Ext                - 2420
=OUTC15  Abs   21537 #05421 - 1581  2163  2211  2850
=OUTEL1  Abs   21248 #05300 - 1299  1778  2165  2909  3210  3289
 OUTEL3  Abs   21306 #0533A - 1334  1305
 OUTEL4  Abs   21287 #05327 - 1323  1316
=OUTELA  Abs   21251 #05303 - 1303  1151  1293  2078
 OUTEOL  Abs   21342 #0535E - 1351   214  1319  1331  1335
=OUTNBC  Abs   21539 #05423 - 1582   254  3208
```

```
=OUTNBS  Abs   21542 #05426 -  1583   272   850  1456
 OUTOVF  Abs   21567 #0543F -  1595  1588
 OUTRP   Ext               -  2467
 Outb,+  Abs   22233 #056D9 -  2469  1107  1901  1927  2214
 Outby+  Abs   22215 #056C7 -  2420  1673  2224
 Outby,  Abs   22236 #056DC -  2470   982  2400  2958
 Outbyt  Abs   22240 #056E0 -  2471   692   761   865  1325
 Outlp+  Abs   22211 #056C3 -  2419   976  2389
 Outrtp  Abs   22229 #056D5 -  2467  2398  2415  3146
 PRRMCK  Abs   22122 #0566A -  2379  3135
 PRGDC1  Abs   22350 #0574E -  2654  2644
 PRMCK+  Abs   22119 #05667 -  2377  3192
 PRMCK1  Abs   22146 #05682 -  2389  2383
 PRMCK3  Abs   22156 #0568C -  2393  2412  2417
 PRMCK4  Abs   22174 #0569E -  2400  2396  3143
 PRMCK5  Abs   22180 #056A4 -  2402  2391
 PRMCK7  Abs   22193 #056B1 -  2410  3199
=PRNTDC  Abs   21584 #05450 -  1661
 PRTDC,  Abs   21608 #05468 -  1671  1679
 PRTDCO  Abs   21612 #0546C -  1673  1688  1706
 PRTDC4  Abs   21658 #0549A -  1705  1682
 PRTDC6  Abs   21665 #054A1 -  1710  1685
=PURGDC  Abs   22341 #05745 -  2646
 R3=D10  Ext               -   914
=RDIZDC  Abs   21492 #053F4 -  1488
=READDC  Abs   21642 #0548A -  1686
=RELJMP  Abs   20551 #05047 -   279
=REMDC   Abs   21330 #05352 -  1343
 REMP05  Ext               -  3154
 REMP10  Ext               -  1343
=RENMDC  Abs   22310 #05726 -  2629
=RESETd  Abs   22801 #05911 -  3327
=RESTDC  Abs   21806 #0552E -  1933  1195  1197  1908
 RNMDC2  Abs   22331 #0573B -  2641  2631
 RNMDC3  Abs   22334 #0573E -  2643  2648
=RNUMDC  Abs   22585 #05839 -  2964
 RTNSET  Ext               -   155
=RUNDC   Abs   21768 #05508 -  1915
 RUNDC+  Abs   21784 #05518 -  1921  1918
 Range   Ext               -  1436
=SECRDC  Abs   22585 #05839 -  2965
 SFLGD5  Abs   21701 #054C5 -  1721  1717
=SFLGDC  Abs   21679 #054AF -  1715
=SKIPDC  Abs   22518 #057F6 -  2900   267
=STATDC  Abs   22770 #058F2 -  3248
=STOPDC  Abs   21251 #05303 -  1301
=SUBDC   Abs   22634 #0586A -  3134
 SUBDC1  Abs   22641 #05871 -  3136  3144
 SUBDC5  Abs   22667 #0588B -  3146  3140
 SUBDC6  Abs   22674 #05892 -  3148  3136
 SUBDC7  Abs   22710 #058B6 -  3158  3150
 TFMD20  Abs   22087 #05647 -  2277  2274
=TRACDC  Abs   21244 #052FC -  1296  1721  2127  3081  3083  3330
 TRNFCK  Ext               -  1353
=TRSFMD  Abs   22049 #05621 -  2268
```

```
 UNDC10  Abs   22797 #0590D -  3289  3250  3252  3329
=UNPRDC  Abs   22786 #05902 -  3286
=USERDC  Abs   22604 #0584C -  3078
 USRDC1  Abs   22604 #0584C -  3079
=VARDC   Abs   21372 #0537C -  1418   977  1097  3248
=VARDC+  Abs   21369 #05379 -  1417   915   967
 VARDC1  Abs   21375 #0537F -  1419  1142
 VARDC2  Abs   21404 #0539C -  1430  1423
 VARDC4  Abs   21421 #053AD -  1435  1432
 VARDC6  Abs   21448 #053C8 -  1445  1437
 VARDC7  Abs   21459 #053D3 -  1448  1443
 VARDC8  Abs   21475 #053E3 -  1453  1450
=WAITd   Abs   21651 #05493 -  1697
=WINDWd  Abs   21651 #05493 -  1698
 XMTADR  Ext                -   455
 d0asci  Abs   20781 #0512D -   590   584
=finda   Abs   22628 #05864 -  3089  3079
 finda+  Abs   22624 #05860 -  3088  1193  2795
 gtexta  Abs   21849 #05559 -  2019  1667  2828
 ldcm20  Abs   21214 #052DE -  1206  1332
 oSPDn2  Abs      14 #0000E -    10   470
 out2t0  Abs   22551 #05817 -  2913  1489  2269
=out2tc  Abs   22554 #0581A -  2914   966  1339
 outbyt  Ext                -  2471  2474  2477  2519  3288
 outela  Abs   21182 #052BE -  1151  1049
 pFILDC  Abs       2 #00002 -    10  2842
 polld+  Ext                -  2840
 sRUNDC  Abs       7 #00007 -    10  1915  1924  1933
 sSSTdc  Abs       1 #00001 -    10   163  1317  1330
 savel#  Ext                -   162
 t!      Ext                -  1198  1534
 t*      Ext                -  1771
 t@      Ext                -  1531
 tALL    Ext                -  1715  2646
 tARRAY  Ext                -  1421
 tBASE   Ext                -  2118
 tCARD   Ext                -  2800
 tCLOCK  Abs 1376751 #501EF -    10  3327
 tCOLON  Ext                -  2808
 tELSE   Ext                -  1314
 tEOL    Ext                -   211  1528
 tERROR  Ext                -  2012
 tFN     Ext                -   909
 tIN     Ext                -  3202
 tINT0   Abs 3015151 #E01EF -    10  2272
 tKEYS   Ext                -  2798
 tLBLRF  Ext                -  1194
 tLBLST  Ext                -   202
 tLINE#  Ext                -   578  1196
 tLITRL  Ext                -  2569  2796
 tMAIN   Ext                -  2804
 tMATH   Abs 3539439 #601EF -    10  1718
 tOFF    Ext                -  3082
 tON     Ext                -  3080
 tPCRD   Abs 4063727 #E01EF -    10  2806  2823
```

```
tPORT   Ext              -  2802
tPRMEN  Ext              -  2394
tPRMST  Ext              -   973  2381
tRFILE  Ext              -  1916
tSEMIC  Ext              -   748
tSVAR   Ext              -  1430
tTAB    Ext              -  1683
tTHEN   Ext              -  1303
tTO     Ext              -  2629
tUSING  Ext              -  1662
tXFN    Ext              -   440
tXWORD  Ext              -   437
```

Input Parameters

  Source file name is SG&LDC::MS

  Listing file name is SG/LDC:TI:ML::-1

  Object file name is SG%LDC:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS   BBBB    &     EEEEE  X   X  DDDD
 2            *    S   S  B   B  & &    E      X   X   D  D
 3            *    S      B   B  & &    E       X X    D  D
 4            *     SSS   BBBB    &     EEEE     X     D  D
 5            *        S  B   B  & & &  E       X X    D  D
 6            *    S   S  B   B  &  &   E      X   X   D  D
 7            *     SSS   BBBB    && &  EEEEE  X   X  DDDD
 8            *
 9                    TITLE  Expression Decompile <831212.1206>
10 05922             ABS    #5922
11           ****************************************************************
12           ****************************************************************
13           **
14           ** Name:(S) EXPRDC  -  Expression Decompile
15           ** Name:(S) EXDCLP  -  Funny function decompile reentry point
16           **
17           ** Category:   DCMUTL
18           **
19           ** Purpose:
20           **      EXPRDC:  Decompile expression lists
21           **      EXDCLP:  This is the point where funny function
22           **        decompile routines should reenter the expression
23           **        decompiler.
24           **
25           ** Entry:
26           **      EXPRDC:
27           **        D0=Output stream pointer
28           **        D1=Input stream pointer
29           **        D(A)=End of avail mem pointer
30           **        A(B)=Contents of MEM(D1)
31           **        P=0
32           **      EXDCLP:
33           **        D1 is current input pointer(past FFN tokenization)
34           **        D0 is current output pointer(past FFN text)
35           **
36           ** Exit:
37           **      D0=Updated output pointer
38           **      D1=Updated input pointer(First unused byte)
39           **      A(B)=First unused token
40           **      Carry clear
41           **      P = 0
42           **
43           ** Calls:      VARDC,MOVED0,RANGE,DRANGE,OUT1TK,OUTNBC,MEMERR
44           **
45           ** Uses.......
46           **  Inclusive: A,B,C,R0,R1,R2,S0,S3,S8,S10,S11,D0,D1
47           **
48           ** Stk lvls:   4
49           **
50           ** Detail:
51           **           R0 = Output pointer @ entry
52           **           R1 = Temporary input pointer, Sign holds text len
53           **           R2 = Function text
54           **
55           **      Explanation of terms used:
```

```
56          **        Nullop -- This a 00 byte which is used to preserve
57          **             a spot in the output stream to insert an
58          **             operator later.  It also is used as a marker
59          **             to help find the spot later.
60          **        Denature -- Once operators have been enclosed in
61          **             parentheses or in a function call, the
62          **             of that operator is no longer of any
63          **             consequence to the rest of the expression.
64          **             To prevent operators so enclosed from affecting
65          **             precedence, they are changed (denatured) in
66          **             such a way that they do not look like operators
67          **             but can be recogized later when the time comes
68          **             to expand the operator token into the text
69          **             that corresponds to the token.
70          **
71          **             The expression decompiler keeps track of
72          **             whether the expression has the form of a
73          **             reference expression.  To do this, it uses two
74          **             status bits, NewVal and OldVal.  Each pass
75          **             through the decompile loop, the NewVal flag
76          **             is copied to the OldVal flag, and the NewVal
77          **             flag set, then if the token being decompiled
78          **             is a variable or an array token, the NewVal
79          **             flag is cleared.  When the loop finally hits
80          **             a token which terminates the expression, the
81          **             OldVal flag will be clear only if the last
82          **             token in the expression is not a variable or an
83          **             array.  This is equivalent to whether the
84          **             expression has the form of a value expression.
85          **             If the token that terminated the expression
86          **             was a call by value token and the OldVal flag
87          **             was left clear, then an extra set of parenthesis
88          **             is placed around the entire expression.  This
89          **             feature is used in SUB decompile.
90          **
91          ** Algorithm:
92          **        Expression decompile converts an RPN string of
93          **        operands, operators and functions to an algebraic
94          **        stream of characters.  The RPN stream is examined
95          **        an item at a time starting at the beginning (lowest
96          **        address).  There are several types of items which
97          **        may be encountered in the stream.  The following
98          **        summarizes what happens for each type:
99          **          Operands -- output a nullop followed by text for
100         **               constant
101         **            Single digit constants
102         **            Integer constants (2-12 digits)
103         **            Floating point constants (1-12 digits)
104         **            String constants (single or double quoted strings)
105         **            Alpha variables
106         **            Alpha-digit variables
107         **            String alpha variables
108         **            String alpha-digit variables
109         **          Monadic operators -- search back for a nullop,
110         **               insert operator token just after nullop,
```

```
111             **              insert parentheses around that area if an
112             **              operator of lower precedence is there.  If
113             **              parentheses were inserted then denature any
114             **              operators enclosed therein.
115             **            Unary minus
116             **            NOT
117             **          Dyadic operators -- search back for a nullop,
118             **              replace this nullop with the operator token,
119             **              insert parentheses around that area if an
120             **              operator of equal or lower precedence is there.
121             **              If parentheses were inserted then denature any
122             **              operator enclosed therein.
123             **              Now search back for another nullop, insert
124             **              parentheses if any operator of lower precedence
125             **              was encountered and denature all operators
126             **              within these parentheses.
127             **            ^
128             **            *
129             **            /
130             **            %
131             **            DIV
132             **            +
133             **            &
134             **            Relops
135             **            AND
136             **            OR
137             **          Functions -- Determine number of parameters.
138             **              If the function has no parameters then treat
139             **              it like an operand.  If it has more than one
140             **              parameter then for each "extra" parameter look
141             **              back for a nullop and replace it with a comma.
142             **              Now search for a nullop and insert the function
143             **              name and a left parenthesis just after it and
144             **              denature all operators between it and the
145             **              end of the output where a closing parenthesis
146             **              is appended.
147             **          Funny Functions -- Output a nullop and the text
148             **              for the function name then call the functions
149             **              "decompile" routine.
150             **          LPRP token -- Output a left and right parernthesis.
151             **
152             **
153             **        When any token other than one of the above is
154             **        encountered, that marks the end of the expression.
155             **        The entire output stream is moved from the beginning
156             **        of available memory to the end of available memory.
157             **        Now the copy back process begins.  The first byte
158             **        of the output stream should be a nullop and is
159             **        ignored.  Each remaining byte in the output
160             **        stream is copied back to the beginning of available
161             **        memory except that operators (and denatured operators)
162             **        are expanded to their full text representation and
163             **        any embedded nullops are converted to commas.
164             **        When a single or double quote is encountered, bytes
165             **        are copied verbatim until the corresponding closing
```

```
166                **      quote is found.
167                **
168                ** History:
169                **
170                **    Date     Programmer          Modification
171                **  --------   ----------  ----------------------------------
172                **  06/24/82   B.S.        Updated documentation
173                **  11/09/82   B.S.        Merged Dummy array decompile
174                **                         into expression decompile
175                **  08/30/83   B.S.        Fixed bug in DNATUR (39-1017(3))
176                **  09/06/83   B.S.        Added to documentation
177                **
178                ********************************************************************
179                ********************************************************************
180                 NegExp EQU    0              Status bit definition
181                 NeedP  EQU    3              Status bit definition
182                 VarDCF EQU    8              Status bit definition
183                 XFNFlg EQU    8              Status bit definition
184                 OldVal EQU    10             Status bit definition
185                 NewVal EQU    11             Status bit definition
186
187                        RDSYMB SB%TAB::MS
188
189 05922 136  =EXPRDC CDOEX
190 05925 134         DO=C                      Copy DO to C
191 05928 108         RO=C                      Save output pointer
192 0592B 84B         ST=0   NewVal
193 0592E 84A  =EXDCLP ST=0   OldVal
194 05931 86B         ?ST=0  NewVal
195 05934 50          GOYES  EXDC05
196 05936 85A         ST=1   OldVal
197 05939 84B   EXDC05 ST=0   NewVal
198 0593C 858         ST=1   VarDCF             Request EXPR DC VAR DC
199 0593F 7000        GOSUB  =VARDC             Decompile var if it is one
200 05943 5AE         GONC   EXDCLP             Was a var found? Yes then loop
201 05946 85B         ST=1   NewVal
202 05949 171         D1=D1+ 2                  Increment pointer
203 0594C 137         CD1EX
204 0594F 135         D1=C                      Copy input pointer to C
205 05952 109         R1=C                      Save input pointer
206 05955 102         R2=A                      Save byte in R2
207 05958 3318  EXDC10 LC(4)  (t-)~(tNOT)       RANGE [NOT,MINUS]
            28
208 0595E 7556        GOSUB  RANGEj             In this range?
209 05962 452         GOC    EXDC30             No, continue
210 05965 3338        LC(4)  (tOR)~(t*)         Precedence RANGE
            D8
211 0596B 7C17        GOSUB  NXTNL+             D1=DO, Find nullop
212 0596F 7A97  EXDC15 GOSUB  APPNDP            Append Rparen if needed
213 05973 7987        GOSUB  INS(               Insert Lparen if needed
214 05977 11A         C=R2                      Get back unary operator token
215 0597A 7647        GOSUB  INSCHR             Insert unop before nullop
216 0597E 119   EXDC20 C=R1                     Get input pointer
217 05981 135         D1=C                      Restore D1
218 05984 69AF        GOTO   EXDCLP
```

```
219                *-
220                *-
221 05988 3308  EXDC30 LC(4)  (tOR)~(t^)     RANGE [Uparrow,or]
          D8
222 0598E 7526         GOSUB  RANGEj         In this range?
223 05992 560          GONC   EXDC31         Yes, then process operator
224 05995 63B0         GOTO   EXDC50         No, continue
225                *-
226                *-
227 05999 3308  EXDC31 LC(4)  (tOR)~(t^)     Precedence RANGE
          D8
228 0599F DE           ACEX   A
229 059A1 80D0         P=C    0              Get lower nibble of token
230 059A5 3D0F         LCHEX  3333779ABCCFF0  Magic constant for prec. RANGE
          FCCB
          A977
          3333
231                *              ||||||||||||||-- 80  ^      Range=8F81
232                *              |||||||||||||--- 8F         Range=8F8F [Unused]
233                *              ||||||||||||---- 8E         Range=8F8F [Unused]
234                *              |||||||||||----- 8D  OR     Range=8F8C
235                *              ||||||||||------ 8C  EXOR   Range=8F8C
236                *              |||||||||------- 8B  AND    Range=8F8B
237                *              ||||||||-------- 8A  RELOP  Range=8F8A
238                *              |||||||--------- 89  &      Range=8F89
239                *              ||||||---------- 88  -      Range=8F87
240                *              |||||----------- 87  +      Range=8F87
241                *              ||||------------ 86  DIV    Range=8F83
242                *              |||------------- 85  %      Range=8F83
243                *              ||-------------- 84  /      Range=8F83
244                *              |--------------- 83  *      Range=8F83
245                *
246 059B5 20           P=     0
247 059B7 A8A          A=C    P
248 059BA D6           C=A    A
249 059BC 7BC6  EXDC33 GOSUB  NXTNL+         D1=D0, Find nullop
250 059C0 7947         GOSUB  APPNDP         Append Rparen if needed
251 059C4 7837         GOSUB  INS(           Insert Lparen if needed
252 059C8 112          A=R2                  Get back binary operator
253 059CB 31A1         LC(2)  (=tRELOP)-#70
254 059CF 906          ?A#C   P
255 059D2 B1           GOYES  EXDC35
256 059D4 111          A=R1
257 059D7 133          AD1EX
258 059DA 1570         C=DAT1 P
259 059DE 133          AD1EX
260 059E1 E4           A=A+1  A
261 059E3 101          R1=A
262 059E6 7AD6         GOSUB  INSCHR
263 059EA 112          A=R2
264 059ED 149   EXDC35 DAT1=A B              Write out operator
265 059F0 3308  EXDC36 LC(4)  (tOR)~(t^)
          D8
266 059F6 137          CD1EX                 Swap pointer into C(A)
267 059F9 10A          R2=C                  Save pt of rparen insertion
```

```
268 059FC 137          CD1EX                 Restore pointer to D1
269 059FF DE           ACEX   A              A(A)=Dummy range,C(A)=token
270 05A01 80D0         P=C    0
271 05A05 3D1F         LCHEX  777799AACFFFF1  Magic constant for prec. RANGE
          FFFC
          AA99
          7777
272             *              |||||||||||||-- 80  ^      Range=8F81
273             *              ||||||||||||--- 8F         Range=8F8F [Unused]
274             *              |||||||||||---- 8E         Range=8F8F [Unused]
275             *              ||||||||||----- 8D  OR     Range=8F8F
276             *              |||||||||------ 8C  EXOR   Range=8F8F
277             *              ||||||||------- 8B  AND    Range=8F8C
278             *              |||||||-------- 8A  RELOP  Range=8F8A
279             *              ||||||--------- 89  &      Range=8F8A
280             *              |||||---------- 88  -      Range=8F89
281             *              ||||----------- 87  +      Range=8F89
282             *              |||------------ 86  DIV    Range=8F87
283             *              ||------------- 85  %      Range=8F87
284             *              |-------------- 84  /      Range=8F87
285             *              --------------- 83  *      Range=8F87
286             *
287 05A15 20           P=     0
288 05A17 A8A          A=C    P
289 05A1A D6           C=A    A
290 05A1C 7476         GOSUB  NXTNUL         Find next nullop
291 05A20 7CD6         GOSUB  INS(           Insert lparen if needed
292 05A24 863          ?ST=0  NeedP          Paren needed?
293 05A27 E1           GOYES  EXDC40         No, continue
294 05A29 11A          C=R2                  Get address for insertion
295 05A2C 136          CDOEX                 DO=Point of insertion
296 05A2F 06           RSTK=C                Save DO
297 05A31 71E6         GOSUB  DNATUR         Denature operator tokens
298 05A35 07           C=RSTK
299 05A37 136          CDOEX                 Restore DO
300 05A3A 135          D1=C                  Set D1 to pt of insertion
301 05A3D 3192         LCASC  \)\
302 05A41 7F76         GOSUB  INSCHR         Insert rparen
303 05A45 683F  EXDC40 GOTO   EXDC20
304 05A49 8F00  EXDC50 GOSBVL =DRANGE        Is it a single digit?
          000
305 05A50 401          GOC    EXDC60         No, continue
306 05A53 F0    EXDC55 ASL    A
307 05A55 F0           ASL    A              Insert nullop before digit
308 05A57 8E00         GOSUBL =OUT2TK        Output nullop and digit
          00
309 05A5D 602F         GOTO   EXDC20
310 05A61 31D1  EXDC60 LC(2)  tFLT1          Constant token limit
311 05A65 9E6          ?A>C   B
312 05A68 60           GOYES  EXDC70
313 05A6A 69A3         GOTO   CONDC          Decompile constant
314 05A6E 7BC7  EXDC70 GOSUB  QUOTCK         Is it a quote?
315 05A72 5F1          GONC   EXDC80         No, then continue
316 05A75 F0           ASL    A
317 05A77 F0           ASL    A              Nullop
```

```
318 05A79 8E00          GOSUBL =OUT2TK        Output nullop and quote
          00
319 05A7F 14B   EXDC75  A=DAT1 B
320 05A82 7C05          GOSUB  Out1Tk         Output byte
321 05A86 171           D1=D1+ 2              Increment input pointer
322 05A89 966           ?A#C   B              Is this the closing quote
323 05A8C 3F            GOYES  EXDC75         No, then loop back
324 05A8E 6F9E  EXDC79  GOTO   EXDCLP
325             *-
326             *-
327 05A92 6FC1  EXDC80  GOTO   EXDC90
328             *-
329             *-
330 05A96 3382  .LPRP   LCASC  \)(\           ASCII ()
          92
331 05A9C 7000          GOSUB  =out2tc
332 05AA0 5DE           GONC   EXDC79         (B.E.T.) goto EXPRDC
333             *-
334             *-
335 05AA3 7615  .DMYAR  GOSUB  B=VAR          Get variable text
336 05AA7 76F4          GOSUB  GET2           Append nullop
337 05AAB 1534          A=DAT1 S              Read argument count
338 05AAF 170           D1=D1+ 1              Skip argument count
339 05AB2 A46           C=C+C  S              Multiply by 2 for byte count
340 05AB5 80DF          P=C    15             P points above variable text
341 05AB9 A99           C=B    WP             Copy text into C
342 05ABC A42           C=C+A  S
343 05ABF A42           C=C+A  S              Add 2 or 4 to nibble count
344 05AC2 A4C           A=A-1  S
345 05AC5 A4C           A=A-1  S              Set carry if only 1 subscript
346 05AC8 4D0           GOC    EXDC92         Skip if only one subscript
347 05ACB 3582          LCASC  \),(\
          C292
348 05AD3 580           GONC   EXDC94
349 05AD6 3382  EXDC92  LCASC  \)(\
          92
350 05ADC 80DF  EXDC94  P=C    15
351 05AE0 0C            P=P+1                 Adjust WP length
352 05AE2 7B55          GOSUB  outnbc         Output Nullop & dummy array txt
353 05AE6 674E          GOTO   EXDCLP
354             *-
355             *-
356 05AEA 14B   .ISUB$  A=DAT1 B              Read in subscript count
357 05AED 119           C=R1                  Get input pointer
358 05AF0 E6            C=C+1  A              Skip past count nibble
359 05AF2 109           R1=C                  Save input pointer
360 05AF5 136           CDOEX
361 05AF8 134           D0=C                  Copy D0 to C
362 05AFB 135           D1=C                  then to D1
363 05AFE A0C           A=A-1  P              Adjust subscript count
364 05B01 1C1   FUND10  D1=D1- 2              Move pointer
365 05B04 14F           C=DAT1 B              Read in byte
366 05B07 96E           ?C#0   B              Is it a nullop?
367 05B0A 7F            GOYES  FUND10         No, loop back
368 05B0C A0C           A=A-1  P              Decrement subscript count
```

```
369 05B0F 4C0          GOC    FUND20        Only 1 parm, then done
370 05B12 31C2         LCASC  \,\           Fill in a comma
371 05B16 14D          DAT1=C B             Write out comma
372 05B19 57E          GONC   FUND10        (B.E.T.) Loop back
373            *-
374            *-
375 05B1C 31D5  FUND20 LCASC  \]\
376 05B20 72A1         GOSUB  outbyt        Write out ]
377 05B24 7EE5         GOSUB  DNATUR        Denature within []
378 05B28 31B5         LCASC  \[\           Pseudo-binary operator
379 05B2C 14D          DAT1=C B
380 05B2F AE0          A=0    B
381 05B32 6DBE         GOTO   EXDC36        Check if parens needed in front
382            *-
383            *-
384 05B36 84B   .ARRAY ST=0   NewVal
385 05B39 550          GONC   .FN+          (B.E.T.)
386            *-
387            *-
388 05B3C 853   .FN    ST=1   NeedP         FN flag
389 05B3F 7A74  .FN+   GOSUB  B=VAR
390 05B43 863          ?ST=0  NeedP         Was it a FN?
391 05B46 51           GOYES  FUND70        No, then skip
392 05B48 7554         GOSUB  GET2
393 05B4C 7154         GOSUB  GET2          Make room for 2 chars
394 05B50 3364         LCASC  \NF\          ASCII FN
          E4
395 05B56 23           P=     3             Pointer for 2 chars
396 05B58 A95          B=C    WP            Copy into B
397 05B5B A46   FUND70 C=C+C  S             Double count
398 05B5E A4E          C=C-1  S             Decrement for WP
399 05B61 AF4          A=B    W             Copy text to A
400 05B64 102          R2=A                 Save text in R2
401 05B67 14B          A=DAT1 B             Read in subscript count
402 05B6A 170          D1=D1+ 1             Increment past subscript count
403 05B6D 137          CD1EX                C(A) = input pointer
404 05B70 109   INSFN-  R1=C                 Save text len & input point in R1
405 05B73 20           P=     0
406 05B75 A0C   INSFUN A=A-1  P             Comma count
407 05B78 452          GOC    INSF20        No args
408 05B7B 136          CDOEX
409 05B7E 134          DO=C                 Copy DO to C
410 05B81 06           RSTK=C               Save old DO
411 05B83 181   INSF10 DO=DO- 2             Move ptr
412 05B86 14E          C=DAT0 B             Read in byte
413 05B89 96E          ?C#0   B             Is it a nullop?
414 05B8C 7F           GOYES  INSF10        No, then loop back
415 05B8E A0C          A=A-1  P             Decrement subscript count
416 05B91 492          GOC    INSF30        Commas all inserted?
417 05B94 31C2         LCASC  \,\           Comma
418 05B98 14C          DAT0=C B             Write out comma
419 05B9B 57E          GONC   INSF10        (B.E.T.) Loop back
420            *-
421            *-
422 05B9E AE0   INSF20 A=0    B
```

```
 423 05BA1 7DE3          GOSUB  Out1Tk
 424 05BA5 7984          GOSUB  INSXFN
 425 05BA9 119           C=R1                  Recall text len
 426 05BAC 80DF          P=C    15             Put text len in P
 427 05BB0 112           A=R2                  Recall text
 428 05BB3 7000          GOSUB  =OUTNBS        Output text
 429 05BB7 66CD          GOTO   EXDC20
 430             *-
 431             *-
 432 05BBB 119   INSF30  C=R1                  Recall len
 433 05BBE 80DF          P=C    15
 434 05BC2 07            C=RSTK                Recall old D0
 435 05BC4 161           D0=D0+ 2              Insert following nullop
 436 05BC7 D5            B=C    A              B=End of source
 437 05BC9 136           CDOEX                 (D0=End of source
 438                                            C=Begin of source)
 439 05BCC E1            B=B-C  A              B=Length to move
 440 05BCE 136           CDOEX
 441 05BD1 134           D0=C                  D0=End of source
 442 05BD4 890           ?P=    0              Is pointer zero?
 443 05BD7 50            GOYES  INSF33         Yes, then skip add (Text len=0)
 444 05BD9 809           C+P+1
 445 05BDC 137   INSF33  CD1EX                 D1=End of dest
 446 05BDF 868           ?ST=0  XFNFlg         Is it an XFN?
 447 05BE2 50            GOYES  INSF35         No, then skip
 448 05BE4 175           D1=D1+ 6              Yes, leave 3 extra bytes
 449                                             for "XFN"
 450 05BE7 171   INSF35  D1=D1+ 2              Leave room for left paren
 451 05BEA 137           CD1EX
 452 05BED 7BB3          GOSUB  ROOM?          Is there enough room?
 453 05BF1 06            RSTK=C                Save new output ptr
 454 05BF3 137           CD1EX                 Put back end of dest
 455 05BF6 8F00          GOSBVL =MOVEDO        Open up space
           000
 456 05BFD 1C1           D1=D1- 2              Point to where Lparen goes
 457 05C00 3182          LCASC  \(\            Lparen
 458 05C04 14D           DAT1=C B              Write out Lparen
 459 05C07 7724          GOSUB  INSXFN         Insert "XFN" if needed
 460 05C0B 119           C=R1                  Recall text len
 461 05C0E 94A           ?C=0   S              Is text len zero?
 462 05C11 D0            GOYES  INSF38         Yes, then skip outputting text
 463 05C13 80DF          P=C    15             Put text len in P
 464 05C17 11A           C=R2                  Recall text
 465 05C1A 1541          DAT0=C WP             Write out text
 466 05C1E 07    INSF38  C=RSTK                Recall new output ptr
 467 05C20 134           D0=C                  Restore new output ptr
 468 05C23 20            P=     0
 469 05C25 7990          GOSUB  OUTRP          Output right paren
 470 05C29 79E4          GOSUB  DNATUR         Denature all in parens
 471 05C2D 605D          GOTO   EXDC20
 472             *-
 473             *-
 474 05C31 8E00  .XFN    GOSUBL =GTEXTX        Get XFN text
           00
 475 05C37 5F0           GONC   FUND86         Found?, then continue
```

```
476                *
477                * Exit from GTEXTX:
478                *   D1 @ XFN number
479                *
480 05C3A 7904             GOSUB  XFNTXT
481 05C3E 173              D1=D1+ 4              Move ptr to arg count
482 05C41 137              CD1EX
483 05C44 109              R1=C                  Save updated input pointer
484 05C47 AC5    FUND86    B=C    S              B(S)=WP len
485 05C4A 119              C=R1                  Recall input pointer
486 05C4D 135              D1=C
487 05C50 E6               C=C+1  A              Skip over arg count
488 05C52 109              R1=C                  Put back updated input pointer
489 05C55 6650             GOTO   FUND98         (B.E.T.) Now read arg count
490                *-
491                *-
492 05C59 AC1    .CMPLX    B=0    S              Text length = zero
493 05C5C 302              LCHEX  2              Two parameters
494 05C5F 525              GONC   FUND99         (B.E.T.)
495                *-
496                *-
497 05C62 3300   EXDC90    LC(4)  =FRange
          00
498 05C68 7B43             GOSUB  RANGEj         Is it a function?
499 05C6C 4F5              GOC    EXD100         No, then keep checking
500 05C6F 848              ST=0   XFNFlg         Initialize:  Not XFN
501 05C72 843              ST=0   NeedP          Initialize:  No parens needed
502 05C75 7000             GOSUB  =finda         Check for special cases
503 05C79 3B               CON(2) tXFN
504 05C7B 6BF              REL(3) .XFN           XFN
505 05C7E D7               CON(2) tARRAY
506 05C80 6BE              REL(3) .ARRAY         Array
507 05C83 C7               CON(2) tFN
508 05C85 7BE              REL(3) .FN            FN
509 05C88 7A               CON(2) tISUB$
510 05C8A 06E              REL(3) .ISUB$         Substring
511 05C8D A7               CON(2) tCMPLX
512 05C8F ACF              REL(3) .CMPLX         Complex
513 05C92 AA               CON(2) tLPRP
514 05C94 20E              REL(3) .LPRP          ()
515 05C97 E7               CON(2) tDMYAR
516 05C99 A0E              REL(3) .DMYAR         Dummy array
517 05C9C 4B               CON(2) tFFN
518 05C9E 731              REL(3) .FFN           Funny function
519 05CA1 00               CON(2) 0
520 05CA3 8E00             GOSUBL =GTEXTM        Get function text
          00
521 05CA9 1C1              D1=D1- 2              Point to arg count range
522 05CAC 14F    FUND98    C=DAT1 B              Read arg count
523 05CAF 102              R2=A                  Save function text in R2
524 05CB2 129    FUND99    CR1EX
525 05CB5 AC9              C=B    S              C(S)=length
526 05CB8 129              CR1EX                 Save text length
527 05CBB A8A              A=C    P              Copy argument count
528 05CBE 66BE             GOTO   INSFUN         Insert function
```

```
529                *-
530                *-
531 05CC2 3192 =OUTRP  LCASC  \)\          Output right paren
532 05CC6 8C00 =outbyt GOLONG =OUTBYT
          00
533                *-
534                *-
535 05CCC 311E  EXD100 LC(2)  =tCVAL
536 05CD0 962          ?A=C   B            Is it a call by value
537 05CD3 50           GOYES  EXD110       Yes, then don't suppress parens
538 05CD5 85A          ST=1   OldVal       No, suppress surrounding parens
539 05CD8 848   EXD110 ST=0   VarDCF       Clear flag for VARDC
540 05CDB DB           C=D    A            C=End of avail mem
541 05CDD 135          D1=C                D1=End of avail mem
542 05CE0 10A          R2=C                Store end of avail mem in R2
543 05CE3 110          A=R0                Get start of buffer
544 05CE6 136          CD0EX
545 05CE9 134          D0=C                Get end of output buffer
546 05CEC E2           C=C-A  A            Calculate length
547 05CEE 8E00         GOSUBL =MOved3      Shift expr to end of mem
          00
548 05CF4 87A          ?ST=1  OldVal       Surrounding parens needed?
549 05CF7 A0           GOYES  CPYBAC       No, then okay
550 05CF9 3182         LCASC  \(\          ASCII paren
551 05CFD 75CF         GOSUB  outbyt       Output leading paren
552 05D01 171   CPYBAC D1=D1+ 2            Skip first nullop
553 05D04 15B3  CPYB10 A=DAT1 4            Read byte (and next byte)
554 05D08 171          D1=D1+ 2            Move past this byte
555 05D0B 137          CD1EX
556 05D0E D7           D=C    A            Move pointer from D1 to D
557 05D10 20           P=     0            Need to have pointer at zero
558 05D12 96C          ?A#0   B            Nullop?
559 05D15 43           GOYES  CPYB40
560 05D17 31C2         LCASC  \,\
561 05D1B AEA   CPYB18 A=C    B            Copy char to A for output
562 05D1E 7072  CPYB20 GOSUB  Out1Tk       Output char to buffer
563 05D22 11A   CPYB22 C=R2                C=End of available mem
564 05D25 8A3          ?C=D   A            At end of memory yet?
565 05D28 A0           GOYES  CPYB30       Yes, then finish up
566 05D2A DF           CDEX   A            Move end of avail mem ptr to D
567 05D2C 135          D1=C                Move pointer back to D1
568 05D2F 54D          GONC   CPYB10       (B.E.T.) Loop back for nxt byte
569                *-
570                *-
571 05D32 87A   CPYB30 ?ST=1  OldVal       Closing paren needed?
572 05D35 60           GOYES  CPYB35       No, then okay
573 05D37 778F         GOSUB  OUTRP        Output trailing paren
574 05D3B 119   CPYB35 C=R1                Get input pointer
575 05D3E 135          D1=C                Restore it to D1
576 05D41 1C1          D1=D1- 2            Back up pointer(Last byte was
577                                            not consumed by EXPR DC)
578 05D44 14B   CPYB38 A=DAT1 B            Re-read byte
579 05D47 03           RTNCC               Return from EXPR DeCompile
580                *-
581                *-
```

```
582 05D49 7C50  CPYB40 GOSUB  CPYLIT        Try to copy a literal
583 05D4D 44D          GOC    CPYB22        Literal copied?  Yes, resume loop
584 05D50 31A7         LCASC  \z\           Limit of ASCII chars
585 05D54 9EA          ?A<=C  B             Is it an ASCII char?
586 05D57 7C           GOYES  CPYB20        Yes, then output the ascii char
587                                         No, then expand token
588 05D59 330F         LCHEX  FFF0          RANGE for denatured operator toks
          FF
589 05D5F 7452         GOSUB  RANGEJ        Is byte in this range
590 05D63 490          GOC    CPY4.1        No, then continue
591 05D66 3107         LCHEX  70            Yes, then
592 05D6A B6A          A=A-C  B             renature token
593 05D6D 3378 CPY4.1  LC(4)  (t-)~(t+)     (Minus),Plus
          28
594 05D73 966          ?A#C   B             Is byte a plus?
595 05D76 A1           GOYES  CPYB41        No, then continue
596 05D78 23           P=     3             Select lower 2 bytes
597 05D7A 912          ?A=C   WP            Is this plus followed by a minus?
598 05D7D 5A           GOYES  CPYB22        Yes, then don't print plus
599 05D7F 30F          LCHEX  F             Is it followed by a
600 05D82 912          ?A=C   WP            denatured minus?
601 05D85 D9           GOYES  CPYB22        Yes, then don't print plus
602 05D87 20           P=     0             Must have pointer at 0
603 05D89 31B2         LCASC  \+\           ASCII plus
604 05D8D 5D8          GONC   CPYB18        (B.E.T) Output byte
605             *-
606             *-
607 05D90 31A8 CPYB41  LC(2)  =tRELOP
608 05D94 966          ?A#C   B             Is it a RELOP?
609 05D97 60           GOYES  CPYB50        No, then okay
610 05D99 E7           D=D+1  A
611 05D9B E7           D=D+1  A             Move pointer past RELOP specifier
612 05D9D 7F34 CPYB50  GOSUB  ARITH         Get text for operator
613 05DA1 7C92         GOSUB  outnbc        Output text
614 05DA5 6C7F         GOTO   CPYB22
615             *-
616             *-
617 05DA9 7094 CPYLIT  GOSUB  QUOTCK        Is it a quote?
618 05DAD 500          RTNNC                Return with carry clear if not
619 05DB0 DF   CPYLI1  CDEX   A
620 05DB2 135          D1=C
621 05DB5 D6           C=A    A
622 05DB7 148          DAT0=A B             Output initial quote
623 05DBA 161          D0=D0+ 2
624 05DBD 14B  CPYLI2  A=DAT1 B             Read next char
625 05DC0 171          D1=D1+ 2             Move pointer past char
626 05DC3 148          DAT0=A B             Write out this char
627 05DC6 161          D0=D0+ 2             Move pointer past char
628 05DC9 966          ?A#C   B             Was this the matching quote?
629 05DCC 1F           GOYES  CPYLI2        No, then loop back
630 05DCE 137          CD1EX
631 05DD1 D7           D=C    A             Copy D1 to D for end of
632                                            copy check
633 05DD3 02           RTNSC                Yes, then return with carry set
634             *-
```

```
635                *-
636 05DD5 8E00     .FFN    GOSUBL =GTEXTX
          00
637 05DDB 4E1              GOC    .FFN10
638 05DDE D2               C=0    A
639 05DE0 72EE             GOSUB  outbyt          Output nullop
640 05DE4 AEA              A=C    B               Restore A(B)
641 05DE7 80DF             P=C    15
642 05DEB 8E00             GOSUBL =OUTNBS
          00
643 05DF1 1C4              D1=D1- 5
644 05DF4 8C00             GOLONG =RELJMP
          00
645                *
646                * Upon arrival at FFN decompile routine,
647                *   R1 and D1 point to FFN length byte,
648                *   D0 points past text of FFN name in output stream,
649                *   R0 points at very beginning of current expression
650                *     and must be preserved.
651
652                *-
653                *-
654 05DFA 7942     .FFN10  GOSUB  XFNTXT
655 05DFE 102              R2=A                   Save function text
656 05E01 173              D1=D1+ 4               Skip past FFN number
657 05E04 D0               A=0    A
658 05E06 14B              A=DAT1 B
659 05E09 137              CD1EX
660 05E0C C2               C=C+A  A               Move pointer past entire FFN code
661 05E0E D0               A=0    A               No arguments
662 05E10 6F5D             GOTO   INSFN-
663                *-
664                *-
665 05E14 D2       CONDC   C=0    A
666 05E16 7CAE             GOSUB  outbyt          Output NULLOP
667 05E1A DA               A=C    A               Copy token back to A
668 05E1C 80D0             P=C    0
669 05E20 137              CD1EX
670 05E23 FE               C=-C-1 A               Position D1 so that when
671 05E25 0C               P=P+1                   a complete register is read in
672 05E27 0C               P=P+1                   the most significant digit will
673 05E29 809              C+P+1                   end up in C(S) with less signif
674 05E2C FA               C=-C   A                digits in successively lower
675 05E2E 135              D1=C                    digits of C
676 05E31 1577             C=DAT1 W               Read in constant
677 05E35 17F              D1=D1+ 16              Skip over constant
678 05E38 AF5              B=C    W               Copy constant to B
679 05E3B 80FF             CPEX   15              C(S)=-significant digits
680 05E3F 21               P=     1
681 05E41 A0C              A=A-1  P               Is it an integer constant?
682 05E44 522              GONC   COND10          No, then process real constant
683 05E47 303              LCHEX  3
684 05E4A A8A              A=C    P
685 05E4D 20               P=     0
686 05E4F 811      COND05  BSLC
```

```
687 05E52 A84          A=B    P
688 05E55 7931         GOSUB  Out1Tk
689 05E59 B46          C=C+1  S
690 05E5C 52F          GONC   COND05
691 05E5F 6ECA         GOTO   EXDCLP
692             *-
693             *-
694 05E63 6FA0  CND40j GOTO   COND40
695             *-
696             *-
697 05E67 1533  COND10 A=DAT1 X
698 05E6B 20           P=     0
699 05E6D 3200         LCHEX  500
          5
700 05E72 9B6          ?A>C   X              Is exponent positive?
701 05E75 E5           GOYES  COND30         No, then process neg exp
702 05E77 840          ST=0   NegExp         Positive exponent
703 05E7A 3211         LCHEX  011
          0
704 05E7F 9B6          ?A>C   X              Can it be displayed w/o exp?
705 05E82 1E           GOYES  CND40j         No, then give scientific display
706 05E84 3103         LC(2)  =a0            ASCII 0
707 05E88 AEE          ACEX   B
708 05E8B 811   COND15 BSLC                  Get next digit ready
709 05E8E A84          A=B    P              Copy make ASCII digit in A(B)
710 05E91 7DF0         GOSUB  Out1Tk         Output digit
711 05E95 05           SETDEC
712 05E97 A6E          C=C-1  B              Decrement exponent
713 05E9A 04    COND20 SETHEX
714 05E9C 581          GONC   COND25         Does DP go here?
715 05E9F B46          C=C+1  S              Yes, but ...
716 05EA2 A4E          C=C-1  S              All sig digs output?
717 05EA5 462          GOC    COND28         Yes, then supress DP
718 05EA8 31E2         LCASC  \.\            No, then output it
719 05EAC 761E         GOSUB  outbyt
720 05EB0 DA           A=C    A              Copy back ASCII digit
721 05EB2 BEA          C=-C   B              C(B)=Some negative number
722 05EB5 B46   COND25 C=C+1  S              Are all sig dgts output?
723 05EB8 52D          GONC   COND15         No, then loop back
724 05EBB D5           B=C    A
725 05EBD A65          B=B+B  B              Is exp negative?
726 05EC0 AC1          B=0    S              Output zeros
727 05EC3 480          GOC    COND28         Yes, then done
728 05EC6 A4E          C=C-1  S              C(S)=F
729 05EC9 41C          GOC    COND15         (B.E.T.) Put in trailing 0's
730 05ECC 172   COND28 D1=D1+ 3              Skip over exponent
731 05ECF 6E5A         GOTO   EXDCLP
732             *-
733             *-
734 05ED3 850   COND30 ST=1   NegExp         Negative exponent
735 05ED6 05           SETDEC
736 05ED8 FC           A=-A-1 A              Complement exponent
737 05EDA 04           SETHEX
738 05EDC 80DF         P=C    15             Get # of signif dgts
739 05EE0 D2           C=0    A
```

```
740 05EE2 80F0          CPEX   0
741 05EE6 B8A           C=-C   P            C=# of significant digits
742 05EE9 C2            C=C+A  A            Sig dgts+(-exponent)
743 05EEB AB5           B=C    X            Move to B(X)
744 05EEE 3221          LCHEX  012
          0
745 05EF3 9B1           ?B>C   X            Can constant be displayed
746 05EF6 D1            GOYES  COND40       No, then use scientific notation
747 05EF8 05            SETDEC
748 05EFA A4E  COND35   C=C-1  S            Increment digits to be displayed
749 05EFD A6C           A=A-1  B            Decrement exponent
750 05F00 480           GOC    COND37       Exit loop if number correct
751 05F03 BF5           BSR    W            Add a leading zero
752 05F06 53F           GONC   COND35       (B.E.T.) Loop back
753             *-
754             *-
755 05F09 3103  COND37  LCASC  \0\
756 05F0D AEE           ACEX   B            A(B)=ASCII digit, C(B)=exp
757 05F10 498           GOC    COND20       (B.E.T.)
758             *-
759             *-                          Display in Scientific notation
760 05F13 811   COND40  BSLC
761 05F16 3103          LCASC  \0\
762 05F1A A09           C=C+B  P
763 05F1D 75AD          GOSUB  outbyt
764 05F21 31E2          LCASC  \.\
765 05F25 7D9D          GOSUB  outbyt
766 05F29 AEA           A=C    B
767 05F2C B46   COND45  C=C+1  S
768 05F2F 4F0           GOC    COND50
769 05F32 811           BSLC
770 05F35 A84           A=B    P
771 05F38 7650          GOSUB  Out1Tk
772 05F3C 5FE           GONC   COND45       (B.E.T.)
773             *-
774             *-
775 05F3F 143   COND50  A=DAT1 A            Read in exponent
776 05F42 3354          LCASC  \+E\
          B2
777 05F48 860           ?ST=0  NegExp       Is exponent negative?
778 05F4B E0            GOYES  COND55       No, then skip
779 05F4D B26           C=C+1  XS
780 05F50 B26           C=C+1  XS           Change ASCII + to -
781 05F53 05            SETDEC
782 05F55 F8            A=-A   A            Complement exponent
783 05F57 04            SETHEX
784 05F59 7000  COND55  GOSUB  =out2tc
785 05F5D 80D0          P=C    0
786 05F61 80F2          CPEX   2
787 05F65 80F0          CPEX   0            Reverse nibbles in C(X)
788 05F69 D5            B=C    A
789 05F6B 20            P=     0
790 05F6D 3103          LCASC  \0\
791 05F71 AEA           A=C    B
792 05F74 822           SB=0
```

```
793 05F77 7C00          GOSUB  COND60
794 05F7B 7800          GOSUB  COND60
795 05F7F 7400          GOSUB  COND60
796 05F83 684F          GOTO   COND28
797 05F87 A84   COND60  A=B    P
798 05F8A BB5           BSR    X
799 05F8D 832           ?SB=0
800 05F90 00            RTNYES
801 05F92 8C00  Out1Tk  GOLONG =OUT1TK
          00
802             *-
803             *-
804 05F98 AE5   GET0    B=C    B
805 05F9B 14B   GET1    A=DAT1 B
806 05F9E 171           D1=D1+ 2
807 05FA1 B46   GET2    C=C+1  S
808 05FA4 BF1           BSL    W
809 05FA7 BF1           BSL    W
810 05FAA 03            RTNCC
811             *-
812             *-
813 05FAC 8BB   ROOM?   ?C<=D  A
814 05FAF 00            RTNYES
815 05FB1 8C00          GOLONG =MEMERR
          00
816             *-
817             *-
818 05FB7 8C00  RANGEj  GOLONG =Range
          00
819             *-
820             *-
821             *
822             * B=VAR:
823             *   Entry:  D1 points to variable name
824             *   Exit:   D1 advanced past name
825             *           B contains variable name
826             *           C(S) = number of chars in name
827             *   Uses:   A(B),B(W),C(W)    1 stack level
828             *
829             *
830 05FBD AF1   B=VAR  B=0    W
831 05FC0 AF2          C=0    W              C(S) must be zero
832 05FC3 74DF         GOSUB  GET1           Get first byte
833 05FC7 31D2         LC(2)  =tSVAR         String
834 05FCB 966          ?A#C   B              Is it a string?
835 05FCE 90           GOYES  B=VAR1         No, then continue
836 05FD0 304          LC(1)  =a$            Change to ASCII $
837 05FD3 71CF         GOSUB  GET0           Store it and advance
838 05FD7 3306 B=VAR1  LC(4)  (tADIG9)~(tADIG0)   RANGE of Alpha Digits
          96
839 05FDD 76DF         GOSUB  RANGEj         Is it in range?
840 05FE1 4D0          GOC    B=VAR2         No, then continue
841 05FE4 3103         LCASC  \0\            ASCII digit
842 05FE8 A86          C=A    P              Change to ASCII digit
843 05FEB 79AF         GOSUB  GET0           Store it and advance
```

```
844 05FEF AE8   B=VAR2 B=A    B              Assume an alpha is left
845 05FF2 03           RTNCC
846           *-
847           *-
```

```
848                     EJECT
849                 ****************************************************************
850                 ****************************************************************
851                 **
852                 ** Name:(S) HXDASC  -  Hex to decimal ASCII conversion
853                 **
854                 ** Category:   DCMUTL
855                 **
856                 ** Purpose:
857                 **      Converts a byte to a 3 character decimal ASCII string.
858                 **      Output string contains leading zero(s) if <100.
859                 **
860                 ** Entry:
861                 **      A(B) contains byte to convert
862                 **
863                 ** Exit:
864                 **      B(15-10) contain 3 ASCII decimal digits
865                 **      P      =  0
866                 **
867                 ** Calls:      None
868                 **
869                 ** Uses.......
870                 **  Inclusive: B(W),A(B),C(S),C(B)
871                 **
872                 ** Stk lvls:   1
873                 **
874                 ** History:
875                 **
876                 **    Date      Programmer              Modification
877                 **  --------   ----------   ----------------------------------
878                 **  09/06/83   B.S.         Added documentation
879                 **
880                 ****************************************************************
881                 ****************************************************************
882 05FF4 2F       =HXDASC P=      15
883 05FF6 303           LC(1)   \0\/16
884 05FF9 20            P=      0
885 05FFB 3146   HXDAS+ LC(2)   100             Constant 100
886 05FFF 7610          GOSUB   HEXA30          Do mod 100 reduction
887 06003 31A0          LC(2)   10              Constant 10
888 06007 7E00          GOSUB   HEXA30          Do mod 10 reduction
889 0600B A88           B=A     P
890 0600E 815           BSRC                    Rotate S
891 06011 BF5           BSR     W
892 06014 AC5           B=C     S               Copy upper nib of ASCII digit
893 06017 03            RTNCC                   into place and return
894                 *-
895                 *-
896 06019 BF5    HEXA30 BSR     W
897 0601C 9E2    HEXA35 ?A<C    B               Is byte greater than mod
898 0601F B0            GOYES   HEXA40          No, then exit loop
899 06021 B6A           A=A-C   B
900 06024 B45           B=B+1   S               Increment counter
901 06027 54F           GONC    HEXA35          (B.E.T.)  Loop back
902 0602A BF5    HEXA40 BSR     W               Shift digit
```

```
903 0602D AC5          B=C    S              Copy upper nib of ASCII digit
904 06030 03           RTNCC                 into place and return
905             *-
906             *-
907 06032 868   INSXFN ?ST=0  XFNFlg          Is it an XFN?
908 06035 00           RTNYES                No, then return
909 06037 3585         LCASC  \NFX\
          64E4
910 0603F 25           P=     5
911 06041 8C00  outnbc GOLONG =OUTNBC
          00
912             *-
913             *-
914             ********************************************************************
915             ********************************************************************
916             **
917             ** Name:    XFNTXT  -  Get text of an XFN or FFN
918             **
919             ** Category:   LOCAL
920             **
921             ** Purpose:
922             **      Gets text of a XFN or FFN in form XFNdddeee.
923             **
924             ** Entry:
925             **      D1 points at DDDD (XFN number)
926             **
927             ** Exit:
928             **      XFNFlg(S8) set
929             **      P      = 0
930             **      Carry clear
931             **
932             ** Calls:      HXDASC,HXDAS+,CMPB#C
933             **
934             ** Uses.......
935             **  Inclusive: A(W),B(W),S8(XFNFlg),C(5-0)
936             **
937             ** Stk lvls:   3
938             **
939             ** History:
940             **
941             **    Date     Programmer             Modification
942             **  --------   ----------   ----------------------------------
943             **  10/25/83   B.S.         Added documentation
944             **
945             ********************************************************************
946             ********************************************************************
947             * D1 points at DDDD  (XFN number)
948 06047 858   XFNTXT ST=1   XFNFlg          Set XFN flag
949 0604A 143          A=DAT1 A               Read XFN number into A(3-0)
950 0604D 73AF         GOSUB  HXDASC          Convert a byte to ASCII decimal
951 06051 BF4          ASR    W
952 06054 BF4          ASR    W
953 06057 70AF         GOSUB  HXDAS+          Convert another byte
954 0605B 3103         LCASC  \0\             See if leading 0 can be
955 0605F 8E00         GOSUBL =CMPB#C         suppressed
```

```
            00
    956 06065 2F            P=      15
    957 06067 30B           LC(1)   11              12 nibs
    958 0606A 491           GOC     XFNTX1          Skip if no zero can be suppressed
    959 0606D BF5           BSR     W
    960 06070 BF5           BSR     W               Shift off leading zero
    961 06073 309           LC(1)   9               10 nibs
    962 06076 965           ?B#C    B               Is there another leading zero?
    963 06079 B0            GOYES   XFNTX1          No, then skip
    964 0607B BF5           BSR     W               Yes,
    965 0607E BF5           BSR     W               then shift off another digit
    966 06081 307           LC(1)   7               which leaves length at 8 nibs
    967 06084 20    XFNTX1  P=      0
    968 06086 AF4           A=B     W               Copy text to A
    969 06089 03            RTNCC
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:    NXTNUL  -  Find next nullop
                ** Name:    NXTNL+  -  Find next nullop
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Scans backward in memory to find next nullop, checking
                **      whether parentheses will be required to preserve
                **      precedence.
                **
                ** Entry:
                **      C(3-0) =  Precedence RANGE to check for
                **      P      =  0
                **      NXTNL+:  D0 = Scan start loc+2
                **      NXTNUL:  D1 = Scan start loc+2
                **
                ** Exit:
                **      P      =  0
                **      D1 points to nullop
                **      Carry clear
                **      S3     =  1 iff parens needed
                **
                ** Calls:      QUOTCK,RANGE
                **
                ** Uses.......
                **  Exclusive: A(B),C(A),D1
                **  Inclusive: A(B),C(A),D1    NXTNL+:  A(A)
                **
                ** Stk lvls:   1
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------   ----------   ----------------------------------
                **  09/06/83   B.S.         Updated documentation
                **
                ****************************************************************
                ****************************************************************
0608B 132    NXTNL+ ADOEX
0608E 130           D0=A                    Copy D0 to A,
06091 131           D1=A                    then to D1
06094 D5     NXTNUL B=C     A               Copy prec range
06096 843           ST=0    NeedP           Clear parens needed flag
06099 1C1    NXTN10 D1=D1- 2                Move pointer
0609C 14B           A=DAT1 B                Read byte
0609F 7A91          GOSUB  QUOTCK           Is it a quote?
060A3 5D0           GONC   NXTN40           No, then okay
                                            Yes, then skip literal
060A6 1C1    NXTN30 D1=D1- 2                Point to next char
060A9 14B           A=DAT1 B                Read next char
060AC 966           ?A#C   B                Is it the ending quote?
```

```
1025 060AF 7F          GOYES  NXTN30        No, then loop back
1026           *                             Yes, then continue
1027 060B1 D9   NXTN40 C=B    A             Load prec range
1028 060B3 700F        GOSUB  RANGEj        Check for parens needed
1029 060B7 450         GOC    NXTN20        No, okay
1030 060BA 853         ST=1   NeedP         Yes, set parens needed flag
1031 060BD 96C  NXTN20 ?A#0   B             Is this a nullop?
1032 060C0 9D          GOYES  NXTN10        No, loop
1033 060C2 03          RTNCC                Yes, return
```

```
1034                    EJECT
1035                ********************************************************************
1036                ********************************************************************
1037                **
1038                ** Name:    INSCHR  -  Insert a character in output buffer
1039                **
1040                ** Category:   LOCAL
1041                **
1042                ** Purpose:
1043                **      Insert a character in the output buffer
1044                **
1045                ** Entry:
1046                **      D1      = Point of insertion-2
1047                **      D0      = Ptr to next available byte
1048                **      C(B)    = Byte to be inserted
1049                **      D(A)    =  Ptr to end of available memory
1050                **
1051                ** Exit:
1052                **      Carry clear
1053                **      P       =  0
1054                **      D0      =  Update ptr to next available byte
1055                **      D1      =  Save as at time of call
1056                **
1057                ** Calls:      MOVED0,ROOM?
1058                **
1059                ** Uses.......
1060                **  Exclusive: A(A),B(A)
1061                **  Inclusive: A(A),B(A),D0
1062                **
1063                ** Stk lvls:   3
1064                **
1065                ** History:
1066                **
1067                **    Date      Programmer              Modification
1068                **  --------    ----------   ---------------------------------
1069                **  09/06/83    B.S.         Updated documentation
1070                **
1071                ********************************************************************
1072                ********************************************************************
1073 060C4 06    INSCHR RSTK=C                  Save contents of C
1074 060C6 173         D1=D1+ 4                 Adjust ptr for len calc
1075 060C9 133         AD1EX                    Get begin source+2
1076 060CC 161         D0=D0+ 2                 Calc end of dest
1077 060CF 136         CD0EX
1078 060D2 134         D0=C                     Put back in D0
1079 060D5 135         D1=C                     D1=End of destination
1080 060D8 181         D0=D0- 2                 D0=End of source
1081 060DB 06          RSTK=C                   Save on stack
1082 060DD D5          B=C    A                 and put copy in B(A)
1083 060DF E8          B=B-A  A                 Calculate length
1084 060E1 77CE        GOSUB  ROOM?             Check for enough room
1085 060E5 8F00        GOSBVL =MOVED0           Ripple the buffer down
           000
1086 060EC 20          P=     0                 MOVEMD leaves pointer undefined
1087 060EE 07          C=RSTK                   Get back end of destination
```

```
1088 060F0 134          D0=C                Restore new output ptr
1089 060F3 07           C=RSTK              Restore contents of C
1090 060F5 1C1          D1=D1- 2            Position pointer for insertion
1091 060F8 14D          DAT1=C B            Insert char in buffer
1092 060FB 1C1          D1=D1- 2            Restore pointer to value at call
1093 060FE 03           RTNCC               Return with carry clear
```

```
1094                    EJECT
1095                ****************************************************************
1096                ****************************************************************
1097                **
1098                ** Name:    INS(    -  Inserts left parenthesis if needed
1099                **
1100                ** Category:   LOCAL
1101                **
1102                ** Purpose:
1103                **      Inserts a left parenthesis in output buffer if needed.
1104                **      Does nothing if not needed.
1105                **
1106                ** Entry:
1107                **      P       =  0
1108                **      S3 set if parens needed
1109                **      D(A)    =  Pointer to end of available memory
1110                **      D0      =  Points to next available byte of memory
1111                **      D1      =  Points to point of insertion-2
1112                **
1113                ** Exit:
1114                **      P       =  0
1115                **      D0      =  Points to new next available byte of
1116                **                 available memory
1117                **      Carry clear
1118                **
1119                ** Calls:      INSCHR
1120                **
1121                ** Uses.......
1122                **  Inclusive: A(A),B(A),C(B),D0
1123                **
1124                ** Stk lvls:   3
1125                **
1126                ** History:
1127                **
1128                **    Date      Programmer              Modification
1129                **  --------    ----------   ----------------------------------
1130                **  09/06/83    B.S.         Updated documentation
1131                **
1132                ****************************************************************
1133                ****************************************************************
1134 06100 863    INS(    ?ST=0  NeedP          Are parens needed?
1135 06103 00           RTNYES                 No, then return
1136 06105 3182         LCASC  \(\
1137 06109 6ABF         GOTO   INSCHR
```

```
1138                    EJECT
1139                ****************************************************************
1140                ****************************************************************
1141                **
1142                ** Name:    DNATUR  -  Denature operator tokens
1143                ** Name:    APPNDP  -  Denature operator tokens
1144                **
1145                ** Category:   LOCAL
1146                **
1147                ** Purpose:
1148                **      APPNDP:  Append a closing parenthesis to output buffer
1149                **               then...
1150                **      DNATUR:  Change the appearance of operator tokens in
1151                **               output buffer so that they don't affect
1152                **               precedence.  (Change from 8x to Fx)
1153                **
1154                ** Entry:
1155                **      APPNDP:  S3 set if parens needed
1156                **        D(A)   =  Pointer to end of available memory
1157                **        P      =  0
1158                **          and ...
1159                **      DNATUR:
1160                **        D0     =  Highest address to denature
1161                **        D1     =  Lowest address to denature+2
1162                **
1163                ** Exit:
1164                **      APPNDP:
1165                **        D0     =  Pointer to new end of output buffer
1166                **          and ...
1167                **      P      =  0
1168                **
1169                ** Calls:      OUTRP
1170                **
1171                ** Uses.......
1172                **  Exclusive: A(A),B(A),C(A)
1173                **  Inclusive: A(A),B(A),C(A)
1174                **
1175                ** Stk lvls:   2
1176                **
1177                ** History:
1178                **
1179                **    Date      Programmer             Modification
1180                **  --------    ----------   ---------------------------------
1181                **  06/09/83    B.S.         Updated documentation
1182                **
1183                ****************************************************************
1184                ****************************************************************
1185 0610D 863   APPNDP ?ST=0  NeedP          Are parens needed?
1186 06110 00           RTNYES               No, then return
1187 06112 7CAB         GOSUB  OUTRP         Append right paren
1188                *
1189                *   Now denature operator tokens between D0 and D1
1190                *
1191 06116 136   DNATUR CDOEX
1192 06119 134          D0=C                 Copy D0 to C,
```

```
1193 0611C D5          B=C    A            then to B
1194 0611E 133         AD1EX
1195 06121 131         D1=A                Copy D1 to A
1196 06124 171  DNAT10 D1=D1+ 2            Move pointer to next byte
1197 06127 133         AD1EX               Get pointer into A
1198 0612A 8B8         ?A>=B  A            Done yes?
1199 0612D C1          GOYES  DNAT20       Yes, then exit loop
1200 0612F 133         AD1EX               Put pointer to byte in D1
1201 06132 14F         C=DAT1 B            Read next byte
1202 06135 6000        GOTO   =DNAT12      Jump to bug fix.  (#39-1017(3))
1203            *      P=C    1            Get high nibble of byte
1204 06139      =DNAT18                    Return here from bug fix.
1205 06139 888         ?P#    8            Is high nibble an 8
1206 0613C 8E          GOYES  DNAT10       No, then loop
1207 0613E 21          P=     1            Point to high nibble
1208 06140 30F         LCHEX  F            Denature nibble from 8 to F
1209 06143 14D         DAT1=C B            Write out denatured nibble
1210 06146 5DD         GONC   DNAT10       (B.E.T.) Loop back
1211 06149 20   DNAT20 P=     0            Must exit with P=0
1212 0614B 01          RTN
```

```
1213                    STITLE Arithmetic Operator Decompile
1214                ****************************************************************
1215                ****************************************************************
1216                **
1217                ** Name:(S) ARITH   -  Get Text For An Arithmetic Operator
1218                **
1219                ** Category:   DCMUTL
1220                **
1221                ** Purpose:
1222                **      Returns text for an arithmetic operator
1223                **
1224                ** Entry:
1225                **      P      =  0
1226                **      A(B)=arithmetic operator
1227                **
1228                ** Exit:
1229                **      P      =  WP length of text of arithmetic operator
1230                **      C(WP)  =  Text for arithmetic operator
1231                **                (First ASCII char is in low C, last in high)
1232                **      Carry clear
1233                **
1234                ** Calls:      None
1235                **
1236                ** Uses.......
1237                **  Inclusive: A(X),C(W),P
1238                **
1239                ** Stk lvls:   0
1240                **
1241                ** History:
1242                **
1243                **    Date     Programmer             Modification
1244                **  --------   ----------   ----------------------------------
1245                **  08/01/82   SA           Wrote routine
1246                **  10/19/82   B.S.         Added documentation
1247                **
1248                ****************************************************************
1249                ****************************************************************
1250 0614D 33E3  GTREQL LCASC  \=>\            Load text for >=
           D3
1251 06153 23          P=     3               GTREQL case can be packed out
1252 06155 03          RTNCC                  Return with carry clear
1253                *-
1254                *-
1255 06157 0C    RELOP1 P=P+1
1256 06159 B92         CSL    WP              Shift previous characters left
1257 0615C 0C          P=P+1
1258 0615E B92         CSL    WP
1259 06161 AE6         C=A    B               Install new character
1260 06164 A6C   RELOP2 A=A-1  B               Create next character
1261 06167 A24   RELOP3 A=A+A  XS              Character into relop text?
1262 0616A 4CE         GOC    RELOP1          Yes, then repeat RELOP1
1263 0616D 92C         ?A#0   XS              Relop text complete?
1264 06170 4F          GOYES  RELOP2          No, then repeat RELOP2
1265 06172 03          RTNCC                  Return with carry clear
1266                *-
```

```
1267                *-
1268 06174 32F36    RELOP   LCHEX   63F             Load "?" and GTREQL predicate
1269 06179 922              ?A=C    XS              GTREQL case?
1270 0617C 1D               GOYES   GTREQL          Yes, then goto GTREQL
1271 0617E AEA              A=C     B
1272 06181 3232D            LCHEX   D23             Load "#" and NOTEQL predicate
1273 06186 2F               P=      15
1274 06188 926              ?A#C    XS              NotEqual case?
1275 0618B CD               GOYES   RELOP3          No, then goto RELOP3
1276 0618D 6C90             GOTO    P=1RTN          Set P to 1, RTN
1277                *-
1278                *-
1279 06191 B02    LOGIC     C=C-A   P               Relational operator?
1280 06194 5FD              GONC    RELOP           Yes, then goto RELOP
1281 06197 B06              C=C+1   P               AND operator?
1282 0619A 453              GOC     AND             Yes, then goto AND
1283 0619D B06              C=C+1   P               EXOR operator?
1284 061A0 4D1              GOC     EXOR            Yes, then goto EXOR
1285 061A3 3702F425         LCASC   \ RO \          Load text for OR
                02
1286 061AD 27     P=7RTN    P=      7
1287 061AF 03               RTNCC                   Return with carry clear
1288                *-
1289                *-
1290 061B1 37E4F445 NOT     LCASC   \ TON\          Load text for NOT
                02
1291 061BB 41F              GOC     P=7RTN
1292                *-
1293                *-
1294 061BE 3B025485 EXOR    LCASC   \ ROXE \        Load text for EXOR
                F425
                02
1295 061CC 2B               P=      11
1296 061CE 03               RTNCC                   Return with carry clear
1297                *-
1298                *-
1299 061D0 39021E4  AND     LCASC   \ DNA \         Load text for AND
                4402
1300 061DC 29     P=9RTN    P=      9
1301 061DE 03               RTNCC
1302                *-
1303                *-
1304 061E0 301    =ARITH    LC(1)   =tNOT
1305 061E3 902              ?A=C    P               Logical NOT?
1306 061E6 BC               GOYES   NOT             Yes, then goto NOT
1307 061E8 302              LC(1)   =t-
1308 061EB 902              ?A=C    P               CHANGESIGN operator?
1309 061EE A2               GOYES   MINUS           Yes, then goto MINUS
1310 061F0 821              XM=0                    Record terminal condition
```

```
1311 061F3 306          LC(1)  =tDIV
1312 061F6 902          ?A=C   P              Is it a DIV operator?
1313 061F9 53           GOYES  DIV            Yes, then goto DIV
1314 061FB 30A          LC(1)  =tRELOP
1315 061FE 98E          ?A>=C  P              Relop or logical operator?
1316 06201 09           GOYES  LOGIC          Yes, then goto LOGIC
1317 06203 30C          LCHEX  C
1318 06206 908          ?A=0   P              Exponentiator?
1319 06209 B0           GOYES  EXPONT         Yes, then goto EXPONT
1320 0620B A86          C=A    P
1321 0620E A06          C=C+C  P
1322 06211 B8A          C=-C   P
1323 06214 80D0 EXPONT  P=C    0
1324 06218 3FD2 MINUS   LCASC  \+.%/*^&-\     Load operator character
             62E5
             A2F2
             52E2
             B2
1325 0622A 21   P=1RTN  P=     1
1326 0622C 03           RTNCC                 Return with carry clear
1327 0622E 3902 DIV     LCASC  \ VID \        Load text for DIV
             4494
             6502
1328 0623A 41A          GOC    P=9RTN         (B.E.T.)  Set P=9, RTN
```

```
1329                    EJECT
1330                ****************************************************************
1331                ****************************************************************
1332                **
1333                ** Name:(S) QUOTCK  -  Quote and Apostrophe Check
1334                **
1335                ** Category:   GENUTL
1336                **
1337                ** Purpose:
1338                **      Checks if A(B) is a quote or an apostrophe
1339                **
1340                ** Entry:
1341                **      P      =  0
1342                **      A(B)   =  Byte to be checked
1343                **
1344                ** Exit:
1345                **      P      =  0
1346                **      Carry set iff A(B) is a quote or an apostrophe
1347                **
1348                ** Calls:      None
1349                **
1350                ** Uses.......
1351                **  Inclusive: C(B)
1352                **
1353                ** Stk lvls:   0
1354                **
1355                ** History:
1356                **
1357                **    Date     Programmer              Modification
1358                **  --------   ----------   ----------------------------------
1359                **  10/19/82   B.S.         Added documentation
1360                **
1361                ****************************************************************
1362                ****************************************************************
1363 0623D 3122    =QUOTCK LC(2)  \"\
1364 06241 962             ?A=C   B
1365 06244 00              RTNYES
1366 06246 307             LC(1)  \'\
1367 06249 962             ?A=C   B
1368 0624C 00              RTNYES
1369 0624E 03              RTNCC
1370                *-
1371 06250                 END
```

.ARRAY	Abs	23350	#05B36	-	384	506			
.CMPLX	Abs	23641	#05C59	-	492	512			
.DMYAR	Abs	23203	#05AA3	-	335	516			
.FFN	Abs	24021	#05DD5	-	636	518			
.FFN10	Abs	24058	#05DFA	-	654	637			
.FN	Abs	23356	#05B3C	-	388	508			
.FN+	Abs	23359	#05B3F	-	389	385			
.ISUB$	Abs	23274	#05AEA	-	356	510			
.LPRP	Abs	23190	#05A96	-	330	514			
.XFN	Abs	23601	#05C31	-	474	504			
AND	Abs	25040	#061D0	-	1299	1282			
APPNDP	Abs	24845	#0610D	-	1185	212	250		
=ARITH	Abs	25056	#061E0	-	1304	612			
B=VAR	Abs	24509	#05FBD	-	830	335	389		
B=VAR1	Abs	24535	#05FD7	-	838	835			
B=VAR2	Abs	24559	#05FEF	-	844	840			
CMPB#C	Ext			-	955				
CND40j	Abs	24163	#05E63	-	694	705			
COND05	Abs	24143	#05E4F	-	686	690			
COND10	Abs	24167	#05E67	-	697	682			
COND15	Abs	24203	#05E8B	-	708	723	729		
COND20	Abs	24218	#05E9A	-	713	757			
COND25	Abs	24245	#05EB5	-	722	714			
COND28	Abs	24268	#05ECC	-	730	717	727	796	
COND30	Abs	24275	#05ED3	-	734	701			
COND35	Abs	24314	#05EFA	-	748	752			
COND37	Abs	24329	#05F09	-	755	750			
COND40	Abs	24339	#05F13	-	760	694	746		
COND45	Abs	24364	#05F2C	-	767	772			
COND50	Abs	24383	#05F3F	-	775	768			
COND55	Abs	24409	#05F59	-	784	778			
COND60	Abs	24455	#05F87	-	797	793	794	795	
CONDC	Abs	24084	#05E14	-	665	313			
CPY4.1	Abs	23917	#05D6D	-	593	590			
CPYB10	Abs	23812	#05D04	-	553	568			
CPYB18	Abs	23835	#05D1B	-	561	604			
CPYB20	Abs	23838	#05D1E	-	562	586			
CPYB22	Abs	23842	#05D22	-	563	583	598	601	614
CPYB30	Abs	23858	#05D32	-	571	565			
CPYB35	Abs	23867	#05D3B	-	574	572			
CPYB38	Abs	23876	#05D44	-	578				
CPYB40	Abs	23881	#05D49	-	582	559			
CPYB41	Abs	23952	#05D90	-	607	595			
CPYB50	Abs	23965	#05D9D	-	612	609			
CPYBAC	Abs	23809	#05D01	-	552	549			
CPYLI1	Abs	23984	#05DB0	-	619				
CPYLI2	Abs	23997	#05DBD	-	624	629			
CPYLIT	Abs	23977	#05DA9	-	617	582			
DIV	Abs	25134	#0622E	-	1327	1313			
DNAT10	Abs	24868	#06124	-	1196	1206	1210		
DNAT12	Ext			-	1202				
=DNAT18	Abs	24889	#06139	-	1204				
DNAT20	Abs	24905	#06149	-	1211	1199			
DNATUR	Abs	24854	#06116	-	1191	297	377	470	
DRANGE	Ext			-	304				

EXD100	Abs	23756	#05CCC	-	535	499					
EXD110	Abs	23768	#05CD8	-	539	537					
EXDC05	Abs	22841	#05939	-	197	195					
EXDC10	Abs	22872	#05958	-	207						
EXDC15	Abs	22895	#0596F	-	212						
EXDC20	Abs	22910	#0597E	-	216	303	309	429	471		
EXDC30	Abs	22920	#05988	-	221	209					
EXDC31	Abs	22937	#05999	-	227	223					
EXDC33	Abs	22972	#059BC	-	249						
EXDC35	Abs	23021	#059ED	-	264	255					
EXDC36	Abs	23024	#059F0	-	265	381					
EXDC40	Abs	23109	#05A45	-	303	293					
EXDC50	Abs	23113	#05A49	-	304	224					
EXDC55	Abs	23123	#05A53	-	306						
EXDC60	Abs	23137	#05A61	-	310	305					
EXDC70	Abs	23150	#05A6E	-	314	312					
EXDC75	Abs	23167	#05A7F	-	319	323					
EXDC79	Abs	23182	#05A8E	-	324	332					
EXDC80	Abs	23186	#05A92	-	327	315					
EXDC90	Abs	23650	#05C62	-	497	327					
EXDC92	Abs	23254	#05AD6	-	349	346					
EXDC94	Abs	23260	#05ADC	-	350	348					
=EXDCLP	Abs	22830	#0592E	-	193	200	218	324	353	691	731
EXOR	Abs	25022	#061BE	-	1294	1284					
EXPONT	Abs	25108	#06214	-	1323	1319					
=EXPRDC	Abs	22818	#05922	-	189						
FRange	Ext			-	497						
FUND10	Abs	23297	#05B01	-	364	367	372				
FUND20	Abs	23324	#05B1C	-	375	369					
FUND70	Abs	23387	#05B5B	-	397	391					
FUND86	Abs	23623	#05C47	-	484	475					
FUND98	Abs	23724	#05CAC	-	522	489					
FUND99	Abs	23730	#05CB2	-	524	494					
GET0	Abs	24472	#05F98	-	804	837	843				
GET1	Abs	24475	#05F9B	-	805	832					
GET2	Abs	24481	#05FA1	-	807	336	392	393			
GTEXTM	Ext			-	520						
GTEXTX	Ext			-	474	636					
GTREQL	Abs	24909	#0614D	-	1250	1270					
HEXA30	Abs	24601	#06019	-	896	886	888				
HEXA35	Abs	24604	#0601C	-	897	901					
HEXA40	Abs	24618	#0602A	-	902	898					
HXDAS+	Abs	24571	#05FFB	-	885	953					
=HXDASC	Abs	24564	#05FF4	-	882	950					
INS(	Abs	24832	#06100	-	1134	213	251	291			
INSCHR	Abs	24772	#060C4	-	1073	215	262	302	1137		
INSF10	Abs	23427	#05B83	-	411	414	419				
INSF20	Abs	23454	#05B9E	-	422	407					
INSF30	Abs	23483	#05BBB	-	432	416					
INSF33	Abs	23516	#05BDC	-	445	443					
INSF35	Abs	23527	#05BE7	-	450	447					
INSF38	Abs	23582	#05C1E	-	466	462					
INSFN-	Abs	23408	#05B70	-	404	662					
INSFUN	Abs	23413	#05B75	-	406	528					
INSXFN	Abs	24626	#06032	-	907	424	459				

LOGIC	Abs	24977	#06191	-	1279	1316							
MEMERR	Ext			-	815								
MINUS	Abs	25112	#06218	-	1324	1309							
MOVED0	Ext			-	455	1085							
MOved3	Ext			-	547								
NOT	Abs	25009	#061B1	-	1290	1306							
NXTN10	Abs	24729	#06099	-	1017	1032							
NXTN20	Abs	24765	#060BD	-	1031	1029							
NXTN30	Abs	24742	#060A6	-	1022	1025							
NXTN40	Abs	24753	#060B1	-	1027	1020							
NXTNL+	Abs	24715	#0608B	-	1012	211	249						
NXTNUL	Abs	24724	#06094	-	1015	290							
NeedP	Abs	3	#00003	-	181	292	388	390	501	1016	1030	1134	
					1185								
NegExp	Abs	0	#00000	-	180	702	734	777					
NewVal	Abs	11	#0000B	-	185	192	194	197	201	384			
OUT1TK	Ext			-	801								
OUT2TK	Ext			-	308	318							
OUTBYT	Ext			-	532								
OUTNBC	Ext			-	911								
OUTNBS	Ext			-	428	642							
=OUTRP	Abs	23746	#05CC2	-	531	469	573	1187					
OldVal	Abs	10	#0000A	-	184	193	196	538	548	571			
Out1Tk	Abs	24466	#05F92	-	801	320	423	562	688	710	771		
P=1RTN	Abs	25130	#0622A	-	1325	1276							
P=7RTN	Abs	25005	#061AD	-	1286	1291							
P=9RTN	Abs	25052	#061DC	-	1300	1328							
=QUOTCK	Abs	25149	#0623D	-	1363	314	617	1019					
RANGEj	Abs	24503	#05FB7	-	818	208	222	498	589	839	1028		
RELJMP	Ext			-	644								
RELOP	Abs	24948	#06174	-	1268	1280							
RELOP1	Abs	24919	#06157	-	1255	1262							
RELOP2	Abs	24932	#06164	-	1260	1264							
RELOP3	Abs	24935	#06167	-	1261	1275							
ROOM?	Abs	24492	#05FAC	-	813	452	1084						
Range	Ext			-	818								
VARDC	Ext			-	199								
VarDCF	Abs	8	#00008	-	182	198	539						
XFNFlg	Abs	8	#00008	-	183	446	500	907	948				
XFNTX1	Abs	24708	#06084	-	967	958	963						
XFNTXT	Abs	24647	#06047	-	948	480	654						
a$	Abs	36	#00024	-	187	836							
a0	Abs	48	#00030	-	187	706							
finda	Ext			-	502								
out2tc	Ext			-	331	784							
=outbyt	Abs	23750	#05CC6	-	532	376	551	639	666	719	763	765	
outnbc	Abs	24641	#06041	-	911	352	613						
t*	Abs	131	#00083	-	187	210							
t+	Abs	135	#00087	-	187	593							
t-	Abs	130	#00082	-	187	207	593	1307					
tADIG0	Abs	96	#00060	-	187	838							
tADIG9	Abs	105	#00069	-	187	838							
tARRAY	Abs	125	#0007D	-	187	505							
tCMPLX	Abs	122	#0007A	-	187	511							
tCVAL	Abs	225	#000E1	-	187	535							

tDIV	Abs	134	#00086	-	187	1311			
tDMYAR	Abs	126	#0007E	-	187	515			
tFFN	Abs	180	#000B4	-	187	517			
tFLT1	Abs	29	#0001D	-	187	310			
tFN	Abs	124	#0007C	-	187	507			
tISUB$	Abs	167	#000A7	-	187	509			
tLPRP	Abs	170	#000AA	-	187	513			
tNOT	Abs	129	#00081	-	187	207	1304		
tOR	Abs	141	#0008D	-	187	210	221	227	265
tRELOP	Abs	138	#0008A	-	187	253	607	1314	
tSVAR	Abs	45	#0002D	-	187	833			
tXFN	Abs	179	#000B3	-	187	503			
t^	Abs	128	#00080	-	187	221	227	265	

Input Parameters

  Source file name is SB&EXD::MS

  Listing file name is SB/EXD:TI:ML::-1

  Object file name is SBXEXD:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    TTTTT  III   &     FFFFF  X   X    1
 2            *      T     I   & &    F      X   X   11
 3            *      T     I   & &    F       X X     1
 4            *      T     I    &     FFFF     X      1
 5            *      T     I   & & &  F       X X     1
 6            *      T     I   &  &   F      X   X    1
 7            *      T    III   && &  F      X   X   111
 8            *
 9                     TITLE ROM 1 Fix Module
10 06250             ABS    #6250
11            *
12            * The following code fixes bug #39-957(1)
13            * The "array required" attribute when used causes the minimum
14            * number of parameters check to be bypassed.  This code replaces
15            * the check for the closing parenthesis which had been in-line
16            * in the expression parser code as well as adding a check for
17            * the minimum number of parameters.  The first two instructions
18            * of this code along with the GOC DUMAER following the call to it
19            * replace the LCASC \)\ and ?A#C B \ GOYES DUMAER  which were
20            * removed from the expression parser code.  The net code change
21            * in the expression parser module is zero.  The code added to thi
22            * module requires 41 nibbles.
23            *
24            * This subroutine returns with carry set if the dummy array isn't
25            * followed by a closing paren or it the minimum number of paramet
26            * have not been specified.  The carry will be clear otherwise.
27 06250 3192 =DUMA80 LCASC  \)\            Right paren required
28 06254 966         ?A#C   B              Is the next char a right paren?
29 06257 00          RTNYES                No, then return with carry set
30 06259 119         C=R1                  Recall exad-2
31 0625C BF6         CSR    W
32 0625F BF6         CSR    W
33 06262 BF6         CSR    W              Get Exad-2 to C(A)
34 06265 136         CDOEX
35 06268 14A         A=DAT0 B              Read the min and max parm count
36 0626B 136         CDOEX
37 0626E F0          ASL    A
38 06270 F0          ASL    A              Shift min count into A(XS)
39 06272 9A0         ?A>B   XS             Has min count been satisfied?
40 06275 00          RTNYES                No, then return with carry set
41 06277 03          RTNCC                 Yes, then return with carry clear
42            *
43            *
44            *
45            * The following is used to fix bug # 39-1000(9).  It sets the upp
46            * digits of the output register to zero.  This will prevent a hig
47            * standby current in the keep-alive RAM port which would occur if
48            * the output register bit 4 was left non-zero while shut down.  T
49            * allow a future cost reduction possible, the highest digit of th
50            * output register is also zeroed.  This is the nibble that contro
51            * the beeper circuit.
52 06279 3210 =OUT=1  LCHEX  001            Activate key row containing ATTN ke
         0
53 0627E 570         GONC   OUT=X          (B.E.T.)  Assume carry clear on ent
54 06281 32F0 =OUT=F  LCHEX  00F            Activate all key rows
```

```
          0
 55 06286 801   OUT=X  OUT=C              Set output register
 56 06289 01           RTN
 57             *
 58             *
 59             *
 60             *
 61             *
 62 0628B DE    =DNAT12 ACEX   A           Put C(A) in A(A) and...
 63 0628D 06            RSTK=C             ...save A(A) on stack
 64 0628F 20            P=     0           Need P=0 for QUOTCK
 65 06291 7000          GOSUB  =QUOTCK     Check if its a quote
 66 06295 4E0           GOC    DNAT16      Yes, then find closing quote
 67 06298 07    DNAT14  C=RSTK
 68 0629A DE            ACEX   A           Restore A(A) from stack
 69 0629C 80D1          P=C    1           Do statement which was
 70                                        replaced to make room for
 71                                        the jump to this bug fix.
 72 062A0 6000          GOTO   =DNAT18     Jump back and continue
 73 062A4 171   DNAT16  D1=D1+ 2           Move to next character
 74 062A7 14F           C=DAT1 B           Read next character
 75 062AA 962           ?A=C   B           Is this the matching quote?
 76 062AD BE            GOYES  DNAT14      Yes, then prepare to return
 77 062AF 54F           GONC   DNAT16      (B.E.T.)
```

```
=DNAT12  Abs   25227 #0628B -    62
 DNAT14  Abs   25240 #06298 -    67    76
 DNAT16  Abs   25252 #062A4 -    73    66    77
 DNAT18  Ext                -    72
=DUMA80  Abs   25168 #06250 -    27
=OUT=1   Abs   25209 #06279 -    52
=OUT=F   Abs   25217 #06281 -    54
 OUT=X   Abs   25222 #06286 -    55    53
 QUOTCK  Ext                -    65
```

Input Parameters

  Source file name is TI&FX1::MS

  Listing file name is TI/FX1:TI:ML::-1

  Object file name is TI%FX1:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS    GGG    &      SSS   Y   Y   SSS
 2            *    S   S  G   G  & &    S   S  Y   Y  S   S
 3            *    S      G      & &    S       Y Y   S
 4            *     SSS   G GGG   &      SSS     Y     SSS
 5            *        S  G   G  & & &      S    Y        S
 6            *    S   S  G   G  &  &   S   S    Y    S   S
 7            *     SSS    GGG    && &   SSS     Y     SSS
 8            *
 9                       TITLE  System Commands: DELETE, AUTO, and CAT
10 063A0                 ABS    #063A0
11                       RDSYMB TI%EQU::MS
12                       RDSYMB SB%RAM::MS
13                       RDSYMB SB%IO::MS
14            *
```

```
15                      STITLE CAT Statement
16                ********************************************************************
17                ********************************************************************
18                **
19                ** Name:    CAT     -  Executes CAT Command
20                ** Name:    CAT100  -  Buffer of Nonreadable Chars to Display
21                ** Name:(S) CATEDT  -  Display CATalog Info on the Current File
22                **
23                ** Category:   STEXEC
24                **
25                ** Purpose:    CAT entry point executes CAT Statement
26                **
27                **             CAT100 sends a buffer of nonreadable characters
28                **             to the display.  It turns off the delay and the
29                **             cursor.  It assumes the buffer is pointed to by
30                **             AVMEMS.
31                **
32                **             CATEDT displays the catalog for current file.
33                **
34                ** Entry:      2 ENTRY POINTS:
35                **              1) CAT     - Execution of CAT command. Expect
36                **                           D0 is past tCAT
37                **              2) CATEDT - Displays CAT info on current file
38                **              3) CAT100 - Buffer pointed to by AVMEMS
39                **
40                ** Exit:       via NXTSTM
41                **
42                ** Calls:      FINDA, FINDF, BF2DSP, FSPECx, POLL, NOSCRL,
43                **             RPTKY, SCRLLR, POPBUF, EDIT80, ROMCHK, ROMFND,
44                **             WSR0-3, EOFLCH, tKYSck, D1=CRS, DSPDLY, EOLXC+,
45                **             C=MAIN, CAT95, ROMF-1
46                **
47                ** Uses:       A-D, D1, D0, R0-R3, STMTR0 (All 16 nibbles), S0
48                **         + all of function scratch, S0-S11 - EXPEXC
49                **
50                ** Detail:     CAT [file name][:<dev id>]|ALL|CARD|keys
51                **
52                ** Stack lvls: CAT    - 7
53                **             CATEDT - 6
54                **             CAT100 - 5
55                **
56                ** History:
57                **    Date     Programmer   Modification
58                **  --------   ----------   ------------
59                **  06/28/82   S.W.         Added documentation
60                **  12/07/82   S.W.         All keys popped out of buffer
61                **
62                ********************************************************************
63                ********************************************************************
64                *
65                ********************************************************************
66                ********************************************************************
67                **
68                ** Name:(S) pCAT    -  Poll for CAT on external device
69                **
```

```
70            ** Category:   POLL
71            **
72            ** Type:       POLL
73            **
74            ** Purpose:
75            **      Handles CAT for files not in MAIN, plug-in memory,
76            **       Independent RAM, or CARD
77            **
78            ** Should poll be "Handled" (return with XM=0)?:
79            **      Yes
80            **
81            ** Meaning of "Handling" Poll (what does code do if handled?):
82            **      Takes over command; exits ready to go to next statement
83            **
84            ** Entry conditions for handler (registers, ST, RAM, etc.):
85            **      Carry clear
86            **      B[A] = Poll number.
87            **      HEX mode.
88            **      P=0.
89            **      File name (if any) in A(W) & R0
90            **       If no file name, then A(W)=0
91            **      Device Specifier in D(S),D(X)
92            **
93            ** Normal exit conditions from handler if handled (ST, RAM,
94            ** registers, etc.):
95            **      Carry clear
96            **      HEX mode.
97            **      XM=0.
98            **      PCADDR intact
99            **
100           ** Normal exit conditions from handler if not handled (ST, RAM,
101           ** registers, etc.):
102           **      Carry clear
103           **      HEX mode.
104           **      XM=1.
105           **
106           ** Error exit conditions from handler
107           **      Carry set.
108           **      HEX mode.
109           **      C[0-3] = Error number.
110           **
111           ** Available subroutine levels:
112           **      7
113           **
114           ** NOTE:
115           **
116           **      --SNAPBF, TRFMBF, LDCSPC
117           **      --LEXPTR.--
118           **
119           ** What registers/RAM may be used if handled?:
120           **      A-D, D0, D1, P
121           **      R0-R4, All of STMT/FN Scratch
122           **      Anything is available which is normally available to
123           **       statements.
124           **      S0-S11
```

```
125                **
126                ** What registers/RAM may be used if not handled?:
127                **      Same as if handled, except can't use R0 (see above)
128                **
129                ** What registers/RAM may be used if error exit
130                **      A-D, D0, D1, P
131                **      Same as if handled (See above)
132                **
133                ** Special memory/pointer considerations (are pointers funny?):
134                **      None
135                **
136                ** Envisioned application(s):
137                **      CAT on TAPE, etc
138                **
139                ** Note:
140                **      If no one responds to POLL, error given is eFSPEC.
141                **
142                **      See pCAT$ for related poll
143                **
144                ** History:
145                **
146                **    Date     Programmer              Modification
147                **  --------   ----------   ----------------------------------
148                **  05/10/83   S.W.         Added new documentation header
149                **
150                ********************************************************************
151                ********************************************************************
152                *
153                *
154 063A0 1F08 =WSR0-3 D1=(5) =S-R0-3
          8F2
155 063A7 1554         DAT1=C S
156 063AB 01           RTN
157                *
158                *
159 063AD B1C3  CATHDR NIBHEX B1C3          Cursor off - want nonreadable chars
160 063B1 0202         NIBASC \  NAME  \
          E414
          D454
          0202
161 063C1 0235         NIBASC \ S TYPE \
          0245
          9505
          5402
162 063D1 0202         NIBASC \  LEN   \
          C454
          E402
          0202
163 063E1 0244         NIBASC \ DATE   \
          1445
          5402
          0202
164 063F1 0245         NIBASC \ TIME P0\
          94D4
          5402
```

```
          05F4
    165 06401 2545        NIBASC \RT\
    166 06405 D0A0        NIBHEX D0A0FF     Cursor off - CRLF - holds for delay
          FF
    167             *
    168 0640B 3100  CATNK LC(2)  =tCARD
    169 0640F 962         ?A=C   B
    170 06412 60          GOYES  CAT*
    171 06414 6981        GOTO   CAT37
    172 06418 AF0   CAT*  A=0    W              No file name
    173 0641B 8D00  catcrd GOVLNG =CATCRD
          000
    174             *
    175 06422 0000        REL(5) =PURGDC
          0
    176 06427 0000        REL(5) =CATP
          0
    177 0642C 8E00 =CAT   GOSUBL =eolxc+        Read in byte & compare
          00
    178 06432 581         GONC   CAT05
    179             *
    180             * DISPLAY CATALOG INFO FOR CURRENT FILE
    181             *
    182 06435 7A44 =CATEDT GOSUB  D1=CRS
    183             * Since no scroll on single file CATalog
    184             *  pop key out of buffer from previous operation
    185 06439 7C41  CAT00 GOSUB  popbuf
    186 0643D 7C02        GOSUB  CAT95
    187             * Need to suppress scroll
    188 06441 8E00        GOSUBL =DSPDLY        Enforce DELAY
          00
    189             *
    190             * CRLF W/CURSOR OFF CAUSES SCROLLING
    191             * CRLF W/CURSOR ON  CLEARS DISPLAY
    192             * HORIZONTAL SCROLLING WILL BE HANDLED IN MAIN LOOP
    193             *
    194             * NO LONGER CLEARING ATTN FLAG
    195             *
    196 06447 6245        GOTO   nxtstm
    197             *
    198 0644B 2F    CAT05 P=     15
    199 0644D 30E         LCHEX  E
    200 06450 20          P=     0
    201 06452 7A4F        GOSUB  WSR0-3
    202             *
    203 06456 3100        LC(2)  =tALL          ALL TOKEN
    204 0645A 962         ?A=C   B
    205 0645D 61          GOYES  CAT10
    206 0645F AC2         C=0    S
    207 06462 1554        DAT1=C S
    208             * CHECK FOR CAT keys
    209 06466 8E00        GOSUBL =tKYSck
          00
    210 0646C 4E9         GOC    CATNK          CARRY=> NOT keys
    211 0646F 6531        GOTO   CAT38
```

```
212                *
213                * CAT ALL IN MAINFRAME
214 06473 8E00  CAT10  GOSUBL =C=MAIN        Set C(A) to MAINST
          00
215 06479 1F67  CAT11  D1=(5) =S-R0-1
          8F2
216 06480 145          DAT1=C A              STORE START OF FILE CHAIN
217                * FIND END OF FILE CHAIN
218 06483 8E56         GOSUBL EOFLC+
          A0
219                *
220 06489 1FB7         D1=(5) =S-R0-2        STORE END OF FILE CHAIN
          8F2
221 06490 145          DAT1=C A
222                *
223                * CAT ALL command
224                *
225 06493 1FDA         D1=(5) CATHDR
          360
226 0649A 8E00         GOSUBL =BF2DSP        SEND HEADER TO DISP
          00
227                *
228 064A0 1F67  CAT16  D1=(5) =S-R0-1
          8F2
229 064A7 143          A=DAT1 A
230 064AA 130          DO=A
231 064AD 14E          C=DAT0 B
232 064B0 96A          ?C=0   B              NULL FILE CHAIN?
233 064B3 A5           GOYES  CATAL?
234 064B5 1C4          D1=D1- 5
235                * This may be a problem 1st time thru, ie if key pressed to soon
236                *  it could be lost
237 064B8 141   CAT17  DAT1=A A              WRITE OUT CURRENT CATALOG PTR
238 064BB 131          D1=A
239 064BE 7B81         GOSUB  CAT95
240 064C2 73C0  CAT18  GOSUB  popbuf         Pop key out of buffer
241 064C6 8F00         GOSBVL =RPTKY
          000
242 064CD 4A0          GOC    CATRPT         Repeat vertical scroll?
243 064D0 8E00         GOSUBL =SCRLLR        TAKE CARE OF HORIZONTAL SCROLLING
          00
244 064D6 D8           B=A    A              SAVE KEY CODE
245                * HANDLE VERTICAL SCROLLING HERE
246 064D8 1F17  CATRPT D1=(5) =S-R0-0
          8F2
247 064DF 143          A=DAT1 A              START OF FILE
248 064E2 DC           ABEX   A              FILE START IN B - KEYCODE IN A
249 064E4 8E00         GOSUBL =FINDA
          00
250 064EA 00           CON(2) =k#DOWN        CURSOR DOWN
251 064EC 520          REL(3) CAT19
252 064EF 00           CON(2) =k#UP          CURSOR UP
253 064F1 440          REL(3) CAT21
254 064F4 00           CON(2) =k#TOP         G CURSOR UP
255 064F6 AAF          REL(3) CAT16
```

```
256 064F9 00          CON(2) =k#BOT        G CURSOR DOWN
257 064FB 490         REL(3) CAT32
258 064FE 55          CON(2) 85            EDIT
259 06500 960         REL(3) CAT30
260 06503 B6          CON(2) 107           F CURSOR DOWN (FOR CAT :PORT)
261 06505 B01         REL(3) CTALLP
262 06508 00          CON(2) 0
263             *
264 0650A 537         GONC   CAT31         (B.E.T.)
265             * Null file chain in mainframe - Is this a CAT ALL?
266 0650D 6201  CATAL? GOTO   CTALLP
267             *
268             * CURSOR DOWN PRESSED
269 06511 D4    CAT19  A=B    A
270 06513 D2           C=0    A
271 06515 3102         LC(2)  =oFLENh       A=ADDRESS OF LAST FILE DISPLAYED
272 06519 CA           A=A+C  A
273 0651B 130          DO=A                 PTR TO FILE LENGTH
274 0651E 142          A=DATO A             FILE LENGTH
275 06521 136          CDOEX
276 06524 CA           A=A+C  A             ADDR OF NEXT FILE IN PROG MEM
277             *
278             * ENSURE AREN'T AT MAINEN
279 06526 130          DO=A
280 06529 14E          C=DATO B
281 0652C 96A          ?C=0   B             AT FILE CHAIN END?
282 0652F 39           GOYES  CAT18         YES
283 06531 668F  cat17  GOTO   CAT17         No=> Disp info on next file
284             *
285             * CURSOR UP
286 06535 D0    CAT21  A=0    A             ADDR OF PREV FILE (INITIALIZED)
287 06537 1B67         DO=(5) =S-R0-1       START OF FILE CHAIN
          8F2
288 0653E 146          C=DATO A
289 06541 8A5   CAT22  ?B#C   A             NOT LAST FILE DISPLAYED?
290 06544 B0           GOYES  CAT25
291             * A(A) IS ADDR OF PREV FILE - ZERO IF JUST DISPLAYED 1ST FILE
292 06546 8AC          ?A#0   A             Previous file in chain?
293 06549 8E           GOYES  cat17
294 0654B 667F         GOTO   CAT18         DISPLAY PREVIOUS FILE
295             * KEEP SEARCHING FOR LAST FILE DISPLAYED
296 0654F 134   CAT25  DO=C
297 06552 16F          DO=DO+ lFNAMh
298 06555 16F          DO=DO+ (oFLENh)-(lFNAMh)
299 06558 142          A=DATO A
300 0655B 136          CDOEX                DO=ADDR OF PREV. FILE
301 0655E C2           C=C+A  A             PTR TO NEXT FILE
302 06560 132          ADOEX                A(A)=ADDR OF PREV FILE
303 06563 134          DO=C
304 06566 5AD          GONC   CAT22         (B.E.T.)
305             * MOVE EDIT POINTER
306 06569 D4    CAT30  A=B    A
307 0656B 131          D1=A                 POINTER TO FILE HEADER
308 0656E 7710         GOSUB  popbuf        Pop key out of buffer
309 06572 85B          ST=1   11            No CATalog
```

```
310 06575 84A          ST=0   10             OK to collapse stacks
311 06578 8E00         GOSUBL =EDIT80
          00
312               * SEND PROMPT - TURN ON CURSOR
313 0657E 8F00  CAT31  GOSBVL =NOSCRL        CLEARS BUFFER - INITIALIZES CURSOR
          000
314 06585 6404         GOTO   nxtstm
315               *
316 06589 8C00  popbuf GOLONG =POPBUF
          00
317               *
318 0658F 1BB7  CAT32  D0=(5) =S-R0-2        END OF FILE CHAIN
          8F2
319 06596 142          A=DAT0 A
320 06599 D8           B=A    A
321 0659B 599          GONC   CAT21          (B.E.T.)
322               *
323               * CAT FOR SPECIFIED FILE
324               *
325               *
326 0659E 70F6  CAT37  GOSUB  FSPECj
327 065A2 4C3          GOC    bserr
328               *
329 065A5 8E00  CAT38  GOSUBL =FINDF+
          00
330 065AB 460          GOC    CAT40          FILE NOT FOUND?
331               *
332               * LEGAL MAINFRAME FILE NAME GIVEN & FILE EXISTS
333               * FILE NAME IS IN B & THE FILE IS POINTED TO BY D1
334               * FILE MAY HAVE BEEN FOUND ON A PORT
335               *
336 065AE 6A8E          GOTO   CAT00
337               *
338 065B2 876   CAT40  ?ST=1  6
339 065B5 A2           GOYES  bserr
340 065B7 94A          ?C=0   S              CAT on CARD/PCRD?
341 065BA 12           GOYES  CAT67
342               * Either external device and/or no file name specified
343 065BC ACB          C=D    S
344 065BF A4E          C=C-1  S
345 065C2 441          GOC    CAT65          CAT :MAIN?
346 065C5 A4E          C=C-1  S
347 065C8 4C1          GOC    CAT70
348               *
349 065CB 72B5  CATEXT GOSUB  poll
350 065CF 60           CON(2) =pCAT
351 065D1 8C00  CAT62  GOLONG =pPOLL2        DEFAULT HANDLER - ILL. FILE SPEC
          00
352               *
353 065D7 6B9E  CAT65  GOTO   CAT10
354               *
355 065DB 6F3E  CAT67  GOTO   catcrd
356               *
357               * FILE NOT FOUND => ERROR
358               *
```

```
359 065DF 8C00  bserr  GOLONG =BSERR
          00
360             *
361 065E5 B67   CAT70  D=D+1  B
362 065E8 571         GONC   CAT82          Port# specified?
363             *
364 065EB 71BD        GOSUB  WSR0-3
365 065EF 8E00        GOSUBL =romchk
          00
366 065F5 593         GONC   CAT90          PORT FOUND?
367             *
368 065F8 6D21  CATE34 GOTO   CAT$14         PORT NOT FOUND
369             *
370 065FC 618F  CAT80  GOTO   CAT31
371             *
372             * CAT ON A SINGLE PORT
373 06600 8E00  CAT82  GOSUBL =ROMF-1        FIND SPECIFIED PORT
          00
374 06606 41F         GOC    CATE34         PORT NOT FOUND?
375             *
376 06609 137   CAT83  CD1EX
377 0660C 6C6E        GOTO   CAT11
378             *
379             * F CURSOR DOWN HIT - ONLY HAS MEANING ON CAT :PORT, CAT ALL
380             *
381 06610 1F08  CTALLP D1=(5) =S-R0-3
          8F2
382 06617 1574        C=DAT1 S
383 0661B 94A         ?C=0   S
384 0661E ED          GOYES  CAT80
385 06620 B46         C=C+1  S
386 06623 5B1         GONC   CATCNT         CAT ALL?
387 06626 8E00  CAT85  GOSUBL =romfnd       CAT :PORT
          00
388 0662C 4FC   CAT87  GOC    CAT80          No more ports => NXTSTM
389 0662F 14B   CAT90  A=DAT1 B
390 06632 968         ?A=0   B              NULL FILE CHAIN?
391 06635 1F          GOYES  CAT85
392             * File chain start in D1
393 06637 7E4F        GOSUB  popbuf         Pop F[v] out of buffer
394 0663B 6DCF        GOTO   CAT83          Output CATalog header
395             *
396 0663F 1554  CATCNT DAT1=C S
397 06643 8E00        GOSUBL =romchk
          00
398 06649 62EF        GOTO   CAT87
399             *
400             ********************************************************************
401             ********************************************************************
402             **
403             ** Name:       CAT95
404             **
405             ** Category:   LOCAL
406             **
407             ** Purpose:
```

```
                        **      Writes out CATalog entry to display
                        **
                        ** Entry:
                        **      P       =  0
                        **      2 entry points:
                        **        1) CAT95  - Sends CATalog entry to display.
                        **                    D1 at file header.
                        **        2) CAT100 - Sends buffer of non-readable characters
                        **                    to display; turns cursor & delay off.
                        **                    Assumes buffer pointed to by AVMEMS.
                        **
                        ** Exit: via CRLFND
                        **      P       =  0
                        **
                        ** Calls:       CAT$20, OBCOLL, AVM2DS
                        **
                        ** Uses.......
                        **              A-D, D1,D0, S0, R1,R2
                        **
                        ** Stk lvls:    5
                        **
                        ** History:
                        **
                        **    Date      Programmer              Modification
                        **  --------    ----------   ----------------------------------
                        **  08/29/83    S.W.         Added documentation header
                        **
                        ***************************************************************
                        ***************************************************************
                        *  If INB2DS called:
                        *  LF clears display buffer; & if cursor is off, CR builds
                        *   display and resets cursor position
                        *
                        * Cursor is off
                        * Send buffer of nonreadable characters to the display
                        * Uses 5 subroutine levels
                        *
445 0664D 840    CAT95  ST=0   0
446 06650 72F0          GOSUB  CAT$20
447 06654 7906          GOSUB  obcoll
448 06658 8E00 =CAT100  GOSUBL =AVM2DS
          00
449 0665E 8C00          GOLONG =CRLFND        CRLF; no DELAY
          00
                        *
                        ***************************************************************
                        ***************************************************************
                        **
                        ** Name:     CAT$     -  CATalog Function
                        ** Name:(S)  CAT$20   -  Build CATalog Information Buffer
                        **
                        ** Category:   FNEXEC
                        **
                        ** Purpose:    CAT$ function returns CATalog information on a
                        **             specific file.
```

```
461             **
462             **          The CAT$20 entry point is used to build a
463             **          buffer of CATalog information.  It is used by
464             **          CAT and CAT$ for the card reader, and the
465             **          mainframe.
466             **
467             ** Entry:    2 ENTRY POINTS:
468             **            1) CAT$   - Entry for execution of CAT$
469             **            2) CAT$20 - Entry for CAT. S0 must be clear to
470             **                        flag that the buffer shouldn't be
471             **                        pushed on the stack.  D1 at file
472             **                        header start.
473             **
474             ** Exit:      BUFFER POINTED TO BY CONTENTS OF 'OUTBS'
475             **
476             ** Calls:     OUTNBS, FLTDH, GETRG+, LOCADR, SAVD0, RSTD0,
477             **            FILXQ$, POLL, FTYPDC, PRT#DC, LDCSET, ROMF-1,
478             **            CAT$70, CAT$80, BLNKC+, AVS=D0, OBCOLL, GETPRO,
479             **            FILSKP, BF2STK, DOOUTB, D1=AVE, D1=CRS, C=MAIN,
480             **            AVE=D1
481             **
482             ** Uses:      A-D, D1,D0, S0, R1,R2 -- CAT$20 entry point
483             **       Inclusive:  All the above + F-R0-0, AVMEME, R3, S7-S11
484             **
485             ** Detail:    FILE LENGTH < 1,048,576 NIBS (DECIMAL)
486             **
487             **          IF numer expr <=0 AND no 2nd parm, then defaults
488             **          to current file.
489             **          REGARDLESS OF ANY SPECIFIED STRING EXPRESSION.
490             **
491             **          If called by CAT, then after return AVMEMS should
492             **          be set to OUTBS via OBCOLL
493             **
494             ** Stack lvls: 4
495             **
496             ** History:
497             **
498             **    Date     Programmer   Modification
499             **  --------   ----------   -------------
500             **  06/28/82   S.W.         Increased documentation
501             **  08/05/82   S.W.         Added code to swap date
502             **                           & time, and to add port#
503             **  10/21/82   S.W.         Calls to AVS=D0 & OBCOLL
504             **  06/10/83   S.W.         Replaced calls to LDCSET &
505             **                           BLANKC with call to BLNKC+
506             **  06/28/83   S.W.         Port# saved in R3 (not on
507             **                           RSTK) before calling GETRG+
508             **
509             ****************************************************************
510             ****************************************************************
511             *
512             ****************************************************************
513             ****************************************************************
514             **
515             ** Name:(S) pCAT$   -  Poll for CAT$ on external device
```

```
516             **
517             ** Category:   POLL
518             **
519             ** Type:       POLL
520             **
521             ** Purpose:
522             **      Creates buffer to execute CAT$ for external device
523             **       (related to pCAT)
524             **
525             ** Should poll be "Handled" (return with XM=0)?:
526             **      Yes
527             **
528             ** Meaning of "Handling" Poll (what does code do if handled?):
529             **      Pushes string on stack; AVMEME points to string header
530             **
531             ** Entry conditions for handler (registers, ST, RAM, etc.):
532             **      Carry clear
533             **      B[A] = Poll number.
534             **      HEX mode.
535             **      P=0.
536             **      SO set
537             **      AVMEME points to string header on stack (string contains
538             **       device name)
539             **      If the string is not null, it has already been reversed
540             **       via REV$ (Characters in mem in ascending order)
541             **      PC (DO) saved in F-R0-0
542             **
543             ** Normal exit conditions from handler if handled (ST, RAM,
544             ** registers, etc.):
545             **      Carry clear
546             **      HEX mode.
547             **      XM=0.
548             **      String on stack, with AVMEME pointing to string header
549             **      F-R0-0 preserved from entry
550             **
551             ** Normal exit conditions from handler if not handled (ST, RAM,
552             ** registers, etc.):
553             **      Carry clear
554             **      HEX mode.
555             **      XM=1.
556             **
557             ** Error exit conditions from handler:
558             **      Carry set.
559             **      HEX mode.
560             **      C[0-3] = Error number.
561             **
562             ** Available subroutine levels:
563             **      5
564             **
565             ** NOTE:
566             **      If poll not handled, eFSPEC generated
567             **
568             **      --SCRTCH, SNAPBF, TRFMBF, LDCSPC,--
569             **      --LEXPTR.--
570             **
```

```
                ** What registers/RAM may be used if handled?:
                **      A-D, D0, D1, P
                **      R0-R4, S7-S11, FN Scratch except F-R0-0
                **
                ** What registers/RAM may be used if not handled?:
                **      A-D, D0, D1, P
                **      Same as if handled (see above)
                **
                ** What registers/RAM may be used if error exit
                **      A-D, D0, D1, P
                **      Same as if handled (see above)
                **
                ** Special memory/pointer considerations (are pointers funny?):
                **      No
                **
                ** Envisioned application(s):
                **      To handle: CAT$(n,":TAPE")
                **
                ** History:
                **
                **    Date      Programmer            Modification
                **   --------   ----------   ---------------------------------
                **   06/17/82   S.W.         Improved documentation
                **   07/07/82   S.W.         Modified code before calling BF2STK
                **                           to reference AVMEME instead of TFORN
                **   07/19/82   S.W.         Push null string on stack when
                **                           positive numeric argument too large
                **                           -- used to error.
                **   10/20/82   S.W.         Replaced call to DDOSET (D0<=AVMEMS)
                **                           with call to LDCSET (D0<=OUTBS)
                **
                **   12/06/82   S.W.         Changed exit conditions for
                **                           CAT poll as per N. Zelle
                **
                **   12/13/82   S.W.         Polls on unrecognized file spec
                **                           Polls on file name (may be device
                **                           name without preceding colon)
                **
                ***************************************************************
                ***************************************************************
                *
                *
06664 4812              NIBHEX 4812             Parm descriptor (numeric [,string])
                *
06668 8F00    =CAT$     GOSBVL =SAVD0           Save PC in F-R0-0
      000
                *
0666F 847               ST=0   7
06672 AC3               D=0    S
06675 A4E               C=C-1  S                Parm count - 1
06678 94A               ?C=0   S                DEVICE SPECIFIER GIVEN?
0667B 91                GOYES  CAT$N            No.
                *
0667D 8E00              GOSUBL =ave=d1          Pt AVMEME at string hdr (for poll)
      00
```

```
624 06683 8E00          GOSUBL =FILXQ$          PARSE DEVICE SPECIFIER
          00
625 06689 546           GONC   CAT$PL           NOT A RECOGNIZED SPECIFIER?
626             * FOUND :CARD, :MAIN, OR :PORT
627 0668C 857           ST=1   7                Null string if arg <=0
628 0668F 97C           ?A#0   W                FILE NAME SPECIFIED?
629 06692 C5            GOYES  CAT$PL           May be device w/o colon
630             *
631             * PRESERVE D(S) - DEVICE DESIGNATOR
632             * D1 POINTS PAST LAST CHARACTER IN STRING
633             *
634 06694 850   CAT$N   ST=1   0                PUSH BUFFER ON STACK
635 06697 DB            C=D    A
636 06699 10B           R3=C                    Save port#
637 0669C 8F00          GOSBVL =GETRG+
          000
638 066A3 11B           C=R3
639 066A6 D7            D=C    A                Restore port#
640 066A8 8E00          GOSUBL =ave=d1          Save D1 in AVMEME (ptr past args)
          00
641             * S7 is clear IFF no 2nd parm given
642             *  so that only if no 2nd parm will arg<=0 give curr file
643 066AE 8E00          GOSUBL =fltdh           ROUND & CONVERT TO HEX
          00
644 066B4 ACB           C=D    S
645 066B7 A4E           C=C-1  S
646 066BA 5B4           GONC   CAT$12           NOT :MAIN?
647             *
648 066BD D8            B=A    A                SAVE #
649 066BF 8E00          GOSUBL =C=MAIN          Set C(A) to MAINST
          00
650             * COUNTER IN B(A) - IF TOO BIG, PUSH NULL STRING ON STACK
651 066C5 CD    CAT$05 B=B-1  A                 Arg<=0 => current file if no
652 066C7 457           GOC    CAT$15            device specified, else null strg
653             * C(A) IS POINTING AT START OF FILE CHAIN
654 066CA 135   CAT$08 D1=C
655 066CD 14B           A=DAT1 B
656 066D0 968           ?A=0   B                At end of file chain?
657 066D3 95            GOYES  CAT$NL           Yes => Null string
658             *
659 066D5 CD            B=B-1  A                DECREMENT COUNTER
660 066D7 4E6           GOC    CAT$20           FOUND SPECIFIED FILE?
661 066DA 8EB3          GOSUBL FILSKP
          80
662 066E0 59E           GONC   CAT$08           (B.E.T.)
663             *
664 066E3 8F15  CAT$P1 GOSBVL =D1=AVE           Position D1 at string header
          681
665 066EA 6D31          GOTO   PCEXPR           Restore D0 and goto EXPR
666             *
667             * Poll for external file specification
668             *
669 066EE 7F84  CAT$PL GOSUB  poll
670 066F2 70            CON(2) =pCAT$
671 066F4 4D0           GOC    CAT$ER           HANDLED & ERRORED?
```

```
672 066F7 831          ?XM=0                  HANDLED & OKAY?
673 066FA 9E           GOYES  CAT$P1
674            * Not handled
675 066FC 3300 =INVFSP LC(4)  =eFSPEC         DEFAULT HANDLER - ILL FILE SPEC
          00
676            * IF HANDLED, BUFFER IS AT OUTBS
677            *
678 06702 6CDE  CAT$ER GOTO   bserr
679            *
680            *   PUSH NULL STRING ON STACK
681            *
682 06706 A4E   CAT$12 C=C-1  S
683 06709 5C2          GONC   ct$crd          :CARD?
684            * MUST BE :PORT - PORT # IN D(B)
685 0670C B67          D=D+1  B
686 0670F 4CE          GOC    INVFSP
687 06712 102          R2=A                   SAVE NUMBER
688 06715 8E00         GOSUBL =ROMF-1
          00
689 0671B 112          A=R2
690 0671E D8           B=A    A               RESTORE NUMBER
691 06720 137          CD1EX                  START OF FILE CHAIN IN C(A)
692 06723 51A          GONC   CAT$05          PORT FOUND?
693            *
694 06726 8C00  CAT$14 GOLONG =DVCNFE         DVC NOT FOUND
          00
695            *
696 0672C 8E00  CAT$NL GOSUBL =LDCSET
          00
697 06732 6BC0         GOTO   CAT$65
698            *
699 06736 8D00  ct$crd GOVLNG =CT$CRD
          000
700            *
701            * IF # IS <=0, rtn info on current file or null string
702            *
703 0673D 877   CAT$15 ?ST=1  7
704 06740 CE           GOYES  CAT$NL          2nd parm specified?
705 06742 7D31         GOSUB  D1=CRS          If not, rtn info on curr file
706            *
707 06746 8E00 =CAT$20 GOSUBL =BLNKC+         Call LDCSET,BLNKC,Copy D0 to B(A)
          00
708            *
709            * BLANK-FILL ENTIRE BUFFER
710            *
711 0674C AFA          A=C    W
712 0674F 2F           P=     15
713 06751 30A          LCHEX  A
714 06754 2F    CAT$30 P=     15
715 06756 8E00         GOSUBL =OUTNBS
          00
716 0675C B46          C=C+1  S
717 0675F 54F          GONC   CAT$30
718            * Hard set AVMEMS in case of POLL
719 06762 8E00         GOSUBL =AVS=D0
```

```
          00
720             *
721 06768 D9            C=B    A
722 0676A 134           D0=C
723             *
724 0676D 158F          A=DAT1 1FNAMh          READ IN FILE NAME
725 06771 158F          DAT0=A 1FNAMh
726 06775 16F           D0=D0+ 1FNAMh          STEP OVER FILE NAME & BLANK
727 06778 161           D0=D0+ 2
728 0677B 7C74          GOSUB  GETPR1          READ & INTERPRET PROTECTION NIB
729 0677F 540           GONC   CAT$32
730 06782 2C            P=     12
731 06784 832   CAT$32  ?SB=0                 UNSECURE?
732 06787 60            GOYES  CAT$35
733 06789 0D            P=P-1
734 0678B 0D            P=P-1
735 0678D 3702  CAT$35  LCASC  \EPS \          NOT PRIVATE?
          3505
          54
736 06797 20            P=     0
737 06799 14C           DAT0=C B
738 0679C 163           D0=D0+ 4               STEP OVER TRAILING BLANK, TOO
739 0679F 7F51          GOSUB  FTYPDC          DECOMPILE FILE TYPE
740 067A3 17F           D1=D1+ (oFLENh)-(1FNAMh)
741 067A6 7021          GOSUB  CAT$80          OUTPUT FILE LENGTH
742 067AA 1C4           D1=D1- (1DATEh)-1      PT TO 2nd BYTE OF DATE
743             *
744             * OUTPUT CREATION DATE & TIME
745             *
746 067AD 71E0          GOSUB  CAT$70          GET mm:dd
747 067B1 24            P=     4
748 067B3 31F2          LCASC  \/\
749 067B7 15C9          DAT0=C 10              WRITE IT OUT
750 067BB 169           D0=D0+ 10              STEP OVER IT
751 067BE 173           D1=D1+ 4               PT TO YEAR IN HEADER
752 067C1 14B           A=DAT1 B
753 067C4 20            P=     0
754 067C6 70E0          GOSUB  CAT$71          GET yy
755 067CA F2            CSL    A
756 067CC BF2           CSL    W
757 067CF 31F2          LCASC  \/\
758 067D3 15C5          DAT0=C 6               WRITE OUT /yy
759 067D7 165           D0=D0+ 6
760             * OUTPUT TIME
761 067DA 1C6           D1=D1- (1DATEh)+(1TIMEh)-3
762 067DD 71B0          GOSUB  CAT$70          PT TO 2ND BYTE OF TIME
763 067E1 15C9          DAT0=C 10              WRITE OUT hh:mm
764 067E5 16B           D0=D0+ 12              STEP OVER hh:mm & blank
765             * OUTPUT PORT# (IF ANY)
766             * IF INPUT BUFFER FROM WHERE CAT BEING BUILT ISN'T ON A
767             * PORT, THEN NO FIELD WILL BE OUTPUT
768 067E8 137           CD1EX
769 067EB 8E00          GOSUBL =LOCADR
          00
770 067F1 4C0           GOC    CAT$65          NOT IN A FILE CHAIN?
```

```
771 067F4 A4F          D=D-1  S
772 067F7 460          GOC    CAT$65          IN THE MAIN FILE CHAIN?
773            * FILE IS ON A PORT
774            * PORT# IN D(1), PORT EXTENDER IN D(0)
775 067FA 7340         GOSUB  PRT#DC
776            *
777 067FE 31FF =CAT$65 LCHEX  FF
778 06802 14C          DAT0=C B               OUTPUT BUFFER TERMINATOR
779            *
780 06805 860          ?ST=0  0               CALLED BY CAT?
781 06808 00           RTNYES
782            * SET-UP FOR CALL TO BF2STK
783 0680A 8F15 =CAT$66 GOSBVL =D1=AVE         Point D1 at stack
          681
784 06811 8E00         GOSUBL =DOOUTB         Position C(A) at start of buffer
          00
785            *
786 06817 8F36         GOSBVL =BF2STK
          681
787 0681E 133          AD1EX                  Save stack pointer
788 06821 7C34         GOSUB  obcoll          Collapse AVMEMS
789            *
790 06825 131          D1=A                   Restore stack pointer
791 06828 7600 =PCEXPR GOSUB  RSTDO           Restore PC from F-R0-0
792 0682C 8C00         GOLONG =EXPRJ
          00
793            *
794            *
795            *
796 06832 1BB9 =RSTDO  D0=(5) =F-R0-0
          8F2
797 06839 146          C=DAT0 A
798 0683C 134          D0=C
799 0683F 01           RTN
800            *
```

```
                        STITLE Port Number Decompile
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:(S) PRT#DC  -  Port# Decompile
                    **
                    ** Category:   DCMUTL
                    **
                    ** Purpose:
                    **      Decompiles a port number
                    **
                    ** Entry:
                    **      P      =  0
                    **      D(1)= Port#, D(0)=Extender#
                    **      DO positioned for output (Next 10 nibs blank-filled)
                    **
                    ** Exit:
                    **      P      =  0
                    **      DO incremented by 10 (past trailing blank)
                    **
                    ** Calls:      HEXDEC, CAT$70
                    **
                    ** Uses.......
                    **  Inclusive: A,B,C,P,DO
                    **
                    ** Stk lvls:   1
                    **
                    ** History:
                    **
                    **    Date      Programmer              Modification
                    **  --------   ----------   ----------------------------------
                    **  08/13/83   S.W.         Added documentation
                    **
                    ****************************************************************
                    ****************************************************************
                    *
06841 D2        =PRT#DC C=0    A
06843 AEB               C=D    B
06846 F6                CSR    A               PORT# IN C(0)
06848 DA                A=C    A
0684A 3303              LCASC  \.0\
      E2
06850 A62               C=A+C  B
06853 90B               ?D=0   P
06856 52                GOYES  PRT#D3          NO PORT EXTENDER#?
06858 15C3              DAT0=C 4               OUTPUT d.
0685C 163               DO=DO+ 4               STEP OVER 'd.'
0685F A8B               C=D    P
06862 A8A               A=C    P
06865 8F00              GOSBVL =HEXDEC
      000
0686C 04                SETHEX
0686E 7830              GOSUB  CAT$71
06872 15C3              DAT0=C 4
06876 165               DO=DO+ 6               STEP OVER 'dd '
```

```
 854 06879 01           RTN
 855             *
 856 0687B 14C   PRT#D3 DAT0=C B              OUTPUT d
 857 0687E 169          D0=D0+ 10             STEP OVER 'd    '
 858 06881 01           RTN
 859             *
 860             ********************************************************************
 861             ********************************************************************
 862             **
 863             ** Name:    D1=CRS  -  Set D1 to contents of CURRST
 864             **
 865             ** Category:   PTRUTL
 866             **
 867             ** Purpose:
 868             **      Points D1 to header of current file
 869             **
 870             ** Entry:
 871             **      CURRST is current
 872             **
 873             ** Exit:
 874             **      D1,C(A) point to start of current file
 875             **
 876             ** Calls:      none
 877             **
 878             ** Uses.......
 879             **  Inclusive: C(A),D1
 880             **
 881             ** Stk lvls:   0
 882             **
 883             ** History:
 884             **
 885             **    Date     Programmer              Modification
 886             **  --------   ----------   ----------------------------------
 887             **  01/25/83   S.W.         Added documentation
 888             **
 889             ********************************************************************
 890             ********************************************************************
 891             *
 892 06883 1FD5 =D1=CRS D1=(5) =CURRST
         5F2
 893 0688A 147          C=DAT1 A
 894 0688D 135          D1=C
 895 06890 01           RTN
 896             *
 897             *
 898             ********************************************************************
 899             ********************************************************************
 900             **
 901             ** Name:    CAT$70  -  Decompiles time and mm/dd for CAT$
 902             **
 903             ** Category:   LOCAL
 904             **
 905             ** Purpose:
 906             **      Decompiles time and mm/dd
 907             **
```

```
908             ** Entry:
909             **      2 entry points:
910             **      CAT$70 - D1 1 nib prior to:
911             **            2nd byte of time (@hr) or date (@month) in file
912             **            header (Decompiles 5 characters)
913             **      CAT$71 - A(B) contains 2 digits to decompile
914             **               P=0
915             **
916             ** Exit:
917             **      P      =  0
918             **      hh:mm or mm:dd in lower 10 nibs of C; In the latter case
919             **      the colon must be written over.
920             **      D1 decremented by 1 from entry
921             **      D0 incremented by 2 from entry (CAT$70 entry only)
922             **
923             ** Calls:      none
924             **
925             ** Uses.......
926             **  Inclusive: A(A), B(A), C, P, D1, D0
927             **
928             ** Stk lvls:   0
929             **
930             ** History:
931             **
932             **    Date     Programmer            Modification
933             **  --------   ----------   ----------------------------------
934             **  08/13/82   S.W.         Added documentation
935             **  11/15/82   S.W.         Code packed
936             **
937             ************************************************************************
938             ************************************************************************
939             *
940 06892 161   CAT$70 D0=D0+ 2              STEP OVER BLANK
941 06895 1573         C=DAT1 X
942 06899 30A          LC(1)  \:\
943 0689C F2           CSL    A
944 0689E F2           CSL    A
945 068A0 1C0          D1=D1- 1
946 068A3 14F          C=DAT1 B
947 068A6 DA           A=C    A
948 068A8 2E           P=     14
949             * WANT LEADING 0'S
950 068AA 3313  CAT$71 LCHEX  3431
          43
951 068B0 20           P=     0
952 068B2 D5           B=C    A              Copy counter & 3 into nib 1
953 068B4 BF2   CAT$75 CSL    W
954 068B7 BF2          CSL    W
955 068BA AE9          C=B    B              Read in 30
956 068BD A86          C=A    P              Read digit into low nib
957 068C0 F4           ASR    A              Get next digit into low nib
958 068C2 A0D          B=B-1  P              decrement counter
959 068C5 5EE          GONC   CAT$75
960 068C8 03           RTNCC
961             *
```

```
962             ****************************************************************
963             ****************************************************************
964             **
965             ** Name:    CAT$80  -  Converts hex# to dec & outputs it
966             **
967             ** Category:   DCMUTL
968             **
969             ** Purpose:
970             **      Converts a hex# to decimal & outputs it, replacing
971             **      leading zeros with blanks
972             **
973             ** Entry:
974             **      P      = 0
975             **      3 entry points:
976             **      1) CAT$80 - File length decompile entry point.
977             **                  D1 positioned at file length in header.
978             **                  C(A) contains offset to data.
979             **                  Decompiles 6 digit number.
980             **      2) CAT$83 - A contains hex number to be decompiled.
981             **                  This number will be divided by 2 &
982             **                  decompiled as P specifies (explained below)
983             **      3) CAT$90 - Decompiles hex number in A.
984             **                  P=0 for up to 6 digits; P=12 for 2 digits;
985             **                  P=14 for 5 digits.
986             **
987             ** Exit:
988             **      P      = 0
989             **      D0 past output number, pointing to a blank
990             **
991             ** Calls:      HEXDEC, D0ASCI
992             **
993             ** Uses.......
994             **  Inclusive: A,B,C,D0
995             **
996             ** Stk lvls:   2
997             **
998             ** History:
999             **
1000            **    Date      Programmer          Modification
1001            **  --------    ----------   ----------------------------------
1002            **  08/13/82    S.W.         Added documentation
1003            **  12/07/82    S.W.         Modified P=14 from 4 to 5 digits
1004            **  06/10/83    S.W.         Add 1 to file length (in bytes)
1005            **                            before converting to nibbles
1006            **                            (rounds up, instead of down).
1007            **                           Also 1st command zeroes out 'M'
1008            **                            field, instead of entire register.
1009            **
1010            ****************************************************************
1011            ****************************************************************
1012            *
1013            *
1014 068CA AD0   CAT$80 A=0    M             ZERO OUT 'CAUSE OF 'ASRB'
1015 068CD 143         A=DAT1 A             FILE LENGTH
1016 068D0 EA          A=A-C  A             AMT OF DATA IN FILE
```

```
1017 068D2 E4          A=A+1  A            Round up
1018           *
1019 068D4 81C  =CAT$83 RSRB               WANT BYTES, NOT NIBS
1020 068D7 8F00 =CAT$90 GOSBVL =HEXDEC
           000
1021 068DE 3559         LCHEX  919495
           4919
1022 068E6 8E00         GOSUBL =DOASCI
           00
1023 068EC 3302  CAT$95 LCASC  \- \
           D2
1024 068F2 14C          DAT0=C B           WRITE OVER FF
1025 068F5 01           RTN
1026           *
1027           *
1028           ********************************************************************
1029           ********************************************************************
1030           **
1031           ** Name:(S) FTYPDC  -  File Type Decompile
1032           **
1033           ** Category:   DCMUTL
1034           **
1035           ** Purpose:
1036           **      Decompiles File Type
1037           **
1038           **      FTYPD+ checks to ensure there's enough memory
1039           **      to output decompiled file type.
1040           **
1041           **      FTYPDC assumes there's enough memory.
1042           **
1043           ** Entry:
1044           **      D0 past a blank (pointing to output buffer)
1045           **      D1 pointing at 4 nibble file type#
1046           **      2 ENTRY POINTS:
1047           **      1) FTYPD+ - D(A) = AVMEME
1048           **      2) FTYPDC - P=0
1049           **
1050           ** Exit:
1051           **      5 character file type written to where D0 pointed;
1052           **      D0 past outputted file type; D1 as it was upon entry
1053           **      Carry clear
1054           **      P=0
1055           **
1056           ** Calls:      FTYPFD, CAT$90, CAT$95, OUTNBS, RDENTY
1057           **
1058           ** Uses:       A-C, D0, R0, P
1059           **
1060           ** Stk lvls:   3
1061           **
1062           ** History:
1063           **
1064           **    Date     Programmer   Modification
1065           **  --------   ----------   ------------
1066           **  10/21/82   S.W.         A=0 W <= A=0 A
1067           **  06/10/83   S.W.         Call CAT$95 to output '-'
```

```
1068                **
1069                ********************************************************************
1070                ********************************************************************
1071                * NOW INTERPRET FILE TYPE
1072 068F7 29      =FTYPD+ P=     9              ENSURE THERE'S ENOUGH MEMORY
1073 068F9 8E00            GOSUBL =OUTNBS
           00
1074 068FF 189             D0=D0- 10
1075                *
1076 06902 8F00    =FTYPDC GOSBVL =FTYPFD
           000
1077 06909 423             GOC    FTYP55          MATCH FOUND?
1078                * UNKNOWN FILE TYPE
1079 0690C D0              A=0    A
1080 0690E 15B3            A=DAT1 4               FILE TYPE#
1081                * SEE IF THE 4-DIGIT SIGNED INTEGER IS POSITIVE
1082 06912 34FF            LC(5)  32767           7FFF HEX
           F70
1083 06919 8BA             ?A<=C  A               POSITIVE?
1084 0691C 31              GOYES  FTYP45
1085 0691E 23              P=     3
1086 06920 B98             A=-A   WP
1087 06923 2E              P=     14
1088 06925 181             D0=D0-  2
1089                *  Negative file# - output minus sign
1090 06928 70CF            GOSUB  CAT$95
1091 0692C 161             D0=D0+ 2               STEP OVER SIGN FIELD
1092 0692F 2E     FTYP45   P=     14              CONVERT HEX TO DEC
1093 06931 72AF            GOSUB  CAT$90          SUPPRESS LEADING 0'S ON
1094 06935 D2              C=0    A
1095 06937 305             LCHEX  5
1096 0693A 03              RTNCC
1097                *
1098 0693C 75E0   FTYP55   GOSUB  RDENTY          READ ENTRY INFORMATION INTO C-REG
1099 06940 BF6             CSR    W               Shift off file type info nib
1100 06943 D0              A=0    A
1101 06945 AEA             A=C    B               A(A) is offset to data
1102 06948 BF6             CSR    W
1103 0694B BF6             CSR    W               Shift off offset to data byte
1104 0694E 15C9            DATO=C 10              WRITE OUT ASCII TEXT
1105 06952 169             D0=D0+ 10              STEP OVER TYPE FIELD
1106 06955 D6              C=A    A               C(A) contains offset to data
1107 06957 03              RTNCC
1108                *
1109                ********************************************************************
1110                ********************************************************************
1111                **
1112                ** Name:    D'LTE   -  DELETE execute
1113                **
1114                ** Category:   STEXEC
1115                **
1116                ** Entry:      D0 past tDELETE
1117                **
1118                ** Exit:       via NXTSTM
1119                **
```

```
1120            ** Calls:      PARMXQ, CHKPSF, FILBLK, D1=CRS, MVMEM+, CLLINK
1121            **             CLPSTK
1122            **
1123            ** Uses:       A-D, D1, D0, R0-R3, S1,S6,S9,S10,
1124            **             1st 5 nibbles of SCRTCH
1125            **
1126            ** Syntax:     DELETE <ALL |<line number>[,<line number>]>
1127            **
1128            ** Detail:     In the case of 'DELETE <line#>, <line#'>, If
1129            **             the line# specified in 1st parm is non-existent,
1130            **             then the 1st line# in program memory which is
1131            **             larger will be substituted.  If the line#
1132            **             specified in the 2nd parm is non-existent, then
1133            **             the largest line# in program memory which is
1134            **             smaller will be substituted.
1135            **
1136            **             The 2nd line# is optional; when it's omitted, the
1137            **             DELETE command is interpreted as a single line
1138            **             DELETE.  In this special case, execution occurs
1139            **             only when a match in line numbers is found.
1140            **
1141            **             CHECKS FOR ILLEGAL PARM RANGE.
1142            **
1143            **             DELETE IS NOT PROGRAMMABLE.
1144            **
1145            ** History:
1146            **
1147            **   Date      Programmer   Modifications
1148            **  --------   ----------   -------------
1149            **  10/13/82   S.W.         Cut code by adding call to GETSTe
1150            **  01/11/83   S.W.         Eliminated poll on non-RAM device
1151            **  01/17/83   S.W.         Added call to CLPSTK
1152            **  03/02/83   J.P.         Packed GETPRe to CHKPSF
1153            **
1154            ************************************************************
1155            ************************************************************
1156            *
1157            *
1158 06959 0000         REL(5) =D'LTP
          0
1159            *
1160 0695E 7522 =D'LTE  GOSUB  PARMXQ
1161            *
1162            * Error exit if:
1163            *   Non BASIC
1164            *   PRIVATE
1165            *   SECURE
1166            *
1167 06962 8E00         GOSUBL =CHKPSF        Check private,secure,filetype
          00
1168            *
1169            * Also Sets D(A)=End of file, D0 at 1st line
1170            *
1171 06968 7A25         GOSUB  FILBLK         FIND BGN & END OF BLK TO DEL
1172 0696C F9           B=-B   A
```

```
1173                * B contains offset
1174 0696E DA              A=C    A              Bgn source
1175 06970 7F0F            GOSUB  D1=CRS         Position C at file header
1176 06974 8E00            GOSUBL =MVMEM+
           00
1177 0697A 451             GOC    D'LTO4         Error? (=> eFACCS)
1178 0697D 8E00            GOSUBL =CLPSTK        Clear SUSP flag, collapse stks
           00
1179 06983 8F00            GOSBVL =CLLINK        Clear links
           000
1180                *
1181 0698A 8C00  nxtstm    GOLONG =NXTSTM
           00
1182 06990 8C00  D'LTO4    GOLONG =MFERR         PROT. ERROR
           00
1183                *
1184                *
1185                *
1186                ******************************************************************
1187                ******************************************************************
1188                **
1189                ** Name:    AUTO    -  AUTO execution
1190                **
1191                ** Category:   STEXEC
1192                **
1193                ** Purpose:    Executes AUTO Statement
1194                **             SYNTAX: AUTO [(line number) [,<increment value>]]
1195                **
1196                ** Entry:      3 ENTRY POINTS:
1197                **                          1) AUTO  - ENTRY POINT FOR INITIAL
1198                **                                     EXECUTION SET-UP.
1199                **                          2) AUTXQ7- ENTRY POINT FOR IMPLICIT
1200                **                                     AUTO EXECUTION.
1201                **
1202                **                          3) AUTLN2- Entry Point for FETCH
1203                **                                     B(A) = Line#
1204                **                                     Displays Line# w/ Cursor
1205                **                                     Carry Clear
1206                **
1207                ** Exit:       via DSPLIN or DCPLIN
1208                **
1209                ** Calls:      PARMXQ, FINDL, LIN#DC, LDCSET, LSTLEN, CHKPSF,
1210                **             SAVELO
1211                **
1212                ** Uses:       A-D, D0, R1, R3, S9,S10, CURRL
1213                **
1214                ** Detail:     AUTO [<line number>] [,<increment value>]
1215                **
1216                ** Stk lvls:   6
1217                **
1218                ** History:
1219                **
1220                **    Date     Programmer   Modification
1221                ** ----------  ----------   ------------
1222                **  06/28/82   S.W.         Added documentation
```

```
1223            **  10/13/82   S.W.         LCHEX OFFFF <= C=0 A/C=C-1 A
1224            **  10/20/82   S.W.         Eliminated L Parm
1225            **  03/07/83   S.W.         Checks prot & file type
1226            **  06/20/83   S.W.         Doesn't add increment if CURRL
1227            **                           is non-existent.
1228            **
1229            ****************************************************************
1230            ****************************************************************
1231            *
1232            *
1233 06996 0000          REL(5) =AUTOP
         0
1234            *
1235 0699B D2   =AUTO    C=0    A
1236 0699D 3101          LCHEX  10
1237 069A1 109           R1=C                    DEFAULT VALUES
1238 069A4 10B           R3=C
1239 069A7 859           ST=1   9                DON'T WANT RANGE CHECKED
1240 069AA 7FE1          GOSUB  PRMXQO
1241 069AE 8E00          GOSUBL =CHKPSF          Error if SECURE,PRIVATE,non-BASIC
         00
1242 069B4 111           A=R1                    BGN DEST.
1243 069B7 11B           C=R3                 INCREMENT VALUE
1244 069BA 1BBC          DO=(5) =AUTINC
         6F2
1245 069C1 15C3          DATO=C 4             WRITE IT OUT TO RAM
1246 069C5 6820          GOTO   AUTLIN        WRITE OUT LINE# FROM A TO DISP BUF
1247            *
1248            * ENTRY PT FOR IMPLICIT AUTO EXECUTION
1249            * AUTO INCREMENT VALUE (LESS 1) IS IN C(A)
1250            * FROM PREVIOUS CALL TO AUTOCK
1251            *
1252 069C9 E6   =AUTXQ7 C=C+1  A
1253 069CB 06           RSTK=C                  Save increment value
1254 069CD 1B8E         DO=(5) =CURRL
         7F2
1255 069D4 8F00         GOSBVL =FINDLR
         000
1256 069DB 07           C=RSTK                  Restore increment
1257 069DD 521          GONC   AUTXQ6           Curr line doesn't exist?
1258            *
1259 069E0 05           SETDEC
1260 069E2 23           P=     3
1261 069E4 A1A          A=A+C  WP               Add increment
1262 069E7 540          GONC   AUTXQ5           ok if P#0, since falls into FINDL
1263            *                               which calls NULLP (sets P=0)
1264 069EA E4           A=A+1  A             WRAP-AROUND
1265 069EC 04    AUTXQ5 SETHEX
1266            *
1267 069EE D8    AUTLIN B=A    A
1268 069F0 8F00  AUTXQ6 GOSBVL =FINDL+          SEE IF LINE# ALREADY EXISTS
         000
1269 069F7 472          GOC    AUTXQ8           Line by that number exists?
1270            *
1271            * Entry for FETCH to Display Line# with Null Line
```

```
1272                *    B = Line#
1273                *
1274                * No line exists by that number
1275 069FA D9      =AUTLN2 C=B    A
1276 069FC 8E00             GOSUBL =savel0        Update Current line
           00
1277 06A02 8E00             GOSUBL =LDCSET        Set up D(A),DO
           00
1278 06A08 8E00             GOSUBL =LIN#A+
           00
1279                * OUTPUT BLANK AFTER LINE #
1280 06A0E 3102             LCHEX  20
1281 06A12 7802             GOSUB  LSTLN+         OUTPUT BLANK & CALC. BUFFER LENGTH
1282 06A16 D4               A=B    A
1283 06A18 8D00             GOVLNG =DSPLIN        SEND LINE TO DISPLAY
           000
1284                *
1285                * SEND READABLE CHARS TO DISPLAY & PUT CURSOR AFTER LINE#
1286                *
1287 06A1F 8C00     AUTXQ8  GOLONG =dcplin        External entry flag set
           00
1288                *
1289                ******************************************************************
1290                ******************************************************************
1291                **
1292                ** Name:    RDENTY  -  Read File Type Entry
1293                **
1294                ** Category:   FILUTL
1295                **
1296                ** Purpose:    Read a file type entry information into C-reg
1297                **
1298                ** Entry:      B(A) = Pointer to start of table entry
1299                **
1300                ** Exit:       C(0) = Execution nibble
1301                **             C(2-1) = Offset to data
1302                **             C(12-3) = ASCII file type name
1303                **             Carry clear
1304                **
1305                ** Uses:       A,C
1306                **
1307                ** History:
1308                **
1309                **    Date     Programmer   Modification
1310                **  --------   ----------   ------------
1311                **  06/28/82   S.W.         Added documentation
1312                **
1313                ******************************************************************
1314                ******************************************************************
1315                *
1316 06A25 D4     =RDENTY A=B    A
1317 06A27 132              ADOEX
1318 06A2A 161              DO=DO+ 2
1319 06A2D 15EC             C=DATO 13
1320 06A31 130              DO=A
1321 06A34 01               RTN
```

```
1322            *
1323
1324
1325
1326            *
1327            *
1328            ****************************************************************
1329            ****************************************************************
1330            **
1331            ** Name:    LIST    -  LIST, PLIST commands
1332            **
1333            ** Category:   STEXEC
1334            **
1335            ** Purpose:    Executes LIST & PLIST
1336            **
1337            ** Entry:      DO past tLIST or tPLIST
1338            **             P= 0
1339            **             2 entry points:
1340            **             1) LIST  - LIST entry point
1341            **             2) LIST* - PLIST entry point.  Requires C(B)
1342            **                        is loaded appropriately for MLFFLG, eg
1343            **                        C(B)=(PRINTt)*16+#F.
1344            **
1345            ** Exit:       via NXTSTM
1346            **
1347            ** Calls:      tKYSck, FSPECx, FINDF, BASKEY, LNARGS, LDCOMP,
1348            **             CK"ON", POLL, D1=CRS, PRPSND, GETPRO, GETST1,
1349            **             PARMXQ, EOLXC+
1350            **
1351            ** Uses:
1352            **   Exclusive..  A-D, D0,D1, R0-R3, S0,S1,S7,S8,S9,S10
1353            **   Inclusive..  A-D, D0,D1, R0-R3, S0,S1,S2,S7,S8,S9,S10
1354            **                STMTR1 (All of it), STMTD0,
1355            **                All of function scratch,
1356            **                S-R0-0 (1st 18 nibs used by CKINFO)
1357            **
1358            ** Detail:     Lists each line for the user-set DELAY time.
1359            **
1360            **             'LIST' IS RESTRICTED TO BASIC & KEY FILES.
1361            **
1362            **             CHecks WIDTH, ATTN KEY AND ANY SERVICE REQUESTS.
1363            **
1364            **             LIST [filename] [line# [,line#]]
1365            **
1366            ** Stk lvls:   6
1367            **
1368            ** History:
1369            **
1370            **    Date      Programmer   Modification
1371            **  --------    ----------   ------------
1372            **  06/28/82    S.W.         Added documentation
1373            **  12/17/82    S.W.         Call CK"ON" instead of CKEXCP;
1374            **                            Check Carry instead of S14
1375            **  03/23/83    S.W.         Added PLIST
1376            **
```

```
1377            ********************************************************************
1378            ********************************************************************
1379            *
1380            ********************************************************************
1381            ********************************************************************
1382            **
1383            ** Name:(S) pLIST   -  Poll for LIST on an external device
1384            **
1385            ** Category:   POLL
1386            **
1387            ** Type:       POLL
1388            **
1389            ** Purpose:
1390            **      LISTS a file on an external device
1391            **
1392            ** Should poll be "Handled" (return with XM=0)?:
1393            **      Yes
1394            **
1395            ** Meaning of "Handling" Poll (what does code do if handled?):
1396            **      Checks protection
1397            **      If file not PRIVATE, LISTS the file, ready to go on to
1398            **      NXTSTM
1399            **
1400            ** Entry conditions for handler (registers, ST, RAM, etc.):
1401            **      B[A] = Poll number.
1402            **      HEX mode.
1403            **      P=0.
1404            **      Blank-filed file name in A(W); R0 contains chars 9 & 10
1405            **      If no file name specified, A=0
1406            **      D(S) contains device id; D(X) contains device address
1407            **
1408            ** Normal exit conditions from handler if handled (ST, RAM,
1409            ** registers, etc.):
1410            **      Carry clear
1411            **      HEX mode.
1412            **      XM=0.
1413            **      PCADDR intact
1414            **
1415            ** Normal exit conditions from handler if not handled (ST, RAM,
1416            ** registers, etc.):
1417            **      Carry clear
1418            **      HEX mode.
1419            **      XM=1.
1420            **
1421            ** Error exit conditions from handler
1422            **      Carry set.
1423            **      HEX mode.
1424            **      C[0-3] = Error number.
1425            **
1426            ** Available subroutine levels:
1427            **      7
1428            **
1429            ** NOTE:
1430            **      For no file name specified, the default error message
1431            **      for 'not handled' will be eFSPEC.
```

```
1432            **
1433            ** What registers/RAM may be used if handled?:
1434            **      A-D, D0, D1, P
1435            **      R0-R4, All Statuses except S13
1436            **      Scratch RAM?
1437            **
1438            ** What registers/RAM may be used if not handled?:
1439            **      A-D, D0, D1, P
1440            **      R1,R2,R3
1441            **      Scratch RAM?
1442            **      Statuses except S13
1443            **  NOTE: R0 MAY NOT BE USED IF NOT HANDLED !!!
1444            **
1445            **
1446            ** What registers/RAM may be used if error exit (POLL only)?:
1447            **      A-D, D0, D1, P
1448            **      R0-R4, Statuses except S13, Scratch Ram
1449            **
1450            ** Envisioned application(s):
1451            **      Listing a file that resides on an external device.
1452            **
1453            ** History:
1454            **
1455            **    Date      Programmer              Modification
1456            **  --------    ----------   -----------------------------------
1457            **  01/01/83    S.W.         Added documentation header to poll
1458            **
1459            ************************************************************************
1460            ************************************************************************
1461            *
1462            ************************************************************************
1463            ************************************************************************
1464            **
1465            ** Name:(S) pLIST2  -  POLL to LIST non-BASIC/non-KEY file type
1466            **
1467            ** Category:   POLL
1468            **
1469            ** Type:       POLL
1470            **
1471            ** Purpose:
1472            **      POLLS to LIST a mainframe file that isn't BASIC or KEY
1473            **
1474            ** Should poll be "Handled" (return with XM=0)?:
1475            **      Yes
1476            **
1477            ** Meaning of "Handling" Poll (what does code do if handled?):
1478            **      LISTs the file on the display device
1479            **      Clears XM
1480            **      Ready to go to NXTSTM
1481            **
1482            ** Entry conditions for handler (registers, ST, RAM, etc.):
1483            **      B[A] = Poll number.
1484            **      HEX mode.
1485            **      P=0.
1486            **      D1 at file header start
```

```
1487            **      A(A) contains file type#
1488            **      DO past file specifier
1489            **
1490            ** Normal exit conditions from handler if handled (ST, RAM,
1491            ** registers, etc.):
1492            **      Carry clear
1493            **      HEX mode.
1494            **      XM=0.
1495            **      Ready to go on to NXTSTM - PCADDR intact
1496            **
1497            ** Normal exit conditions from handler if not handled (ST, RAM,
1498            ** registers, etc.):
1499            **      Carry clear
1500            **      HEX mode.
1501            **      XM=1.
1502            **
1503            ** Error exit conditions from handler
1504            **      Carry set.
1505            **      HEX mode.
1506            **      C[0-3] = Error number.
1507            **
1508            ** Available subroutine levels:
1509            **      7
1510            **
1511            **      STMT/FN Scratch, SCRTCH, SNAPBF, TRFMBF, LDCSPC,
1512            **      LEXPTR.
1513            **
1514            ** What registers/RAM may be used if handled?:
1515            **      A-D, D0, D1, P always available
1516            **      R0-R4, ST, scratch RAM
1517            **
1518            ** What registers/RAM may be used if not handled?:
1519            **      A-D, D0, D1, P always available
1520            **      R0-R4, ST, scratch RAM
1521            **
1522            ** What registers/RAM may be used if error exit (POLL only)?:
1523            **      A-D, D0, D1, P always available
1524            **      R0-R4, ST, scratch RAM
1525            **
1526            ** Envisioned application(s):
1527            **      LISTing files of types other than BASIC and KEY, eg
1528            **      perhaps TEXT or DATA files.
1529            **
1530            ** Default:
1531            **      If POLL not handled, error is Invalid File Type
1532            **
1533            ** History:
1534            **
1535            **    Date     Programmer              Modification
1536            **  --------   ----------   ------------------------------------
1537            **  04/04/83   S.W.         Documented poll
1538            **
1539            ****************************************************************
1540            ****************************************************************
1541            *
```

```
1542 06A36 0000          REL(5) =LISTDC
           0
1543 06A3B 0000          REL(5) =LISTP
           0
1544               *
1545 06A40 31F1 =PLIST   LC(2)  (PRINTt)*16+#F
1546 06A44 6530          GOTO   LIST*
1547               *
1548               * LIST OF NON-MAINFRAME FILE
1549 06A48 876    LSTEXT ?ST=1  6
1550 06A4B C0            GOYES  LSTER
1551 06A4D 7031          GOSUB  poll
1552 06A51 C0            CON(2) =pLIST
1553 06A53 6D7B   LIST30 GOTO   CAT62          DEFAULT HANDLER - ILL FILE SPEC
1554               *
1555 06A57 678B   LSTER  GOTO   bserr
1556               * LIST ON WEIRD FILE TYPE - TYPE POINTED TO BY D1
1557               * D0 IS PROGRAM COUNTER (POINTS TO PARMS, IF ANY)
1558               *
1559 06A5B 7221   LSTPOL GOSUB  poll
1560 06A5F E2            CON(2) =pLIST2
1561               * DEFAULT HANDLER LOADS 'ILLEGAL FILE TYPE' ERROR
1562 06A61 6E12          GOTO   pPOLL3
1563               *
1564 06A65 8D00   LSTXKY GOVLNG =LSTKEY
           000
1565               *
1566 06A6C 0000          REL(5) =LISTDC
           0
1567 06A71 0000          REL(5) =LISTP
           0
1568               *
1569 06A76 31F0 =LIST    LC(2)  (DISPt)*16+#F
1570 06A7A 1F07 =LIST*   D1=(5) =MLFFLG        Write out to MLFFLG
           8F2
1571 06A81 14D           DAT1=C B
1572 06A84 7BFD          GOSUB  D1=CRS
1573 06A88 8E00          GOSUBL =eolxc+        Read in 1 byte & compare
           00
1574 06A8E 442           GOC    LIST50         AT EOL?
1575 06A91 3100          LC(2)  =tCOMMA
1576 06A95 962           ?A=C   B
1577 06A98 B1            GOYES  LIST50
1578               *
1579 06A9A 8E00          GOSUBL =tKYSck
           00
1580 06AA0 590           GONC   LIST20         tKEYS FOUND?
1581               *
1582 06AA3 7BE1   LIST10 GOSUB  FSPECj
1583 06AA7 4FA           GOC    LSTER
1584               *
1585 06AAA 8E00   LIST20 GOSUBL =FINDF+        LOCATE FILE
           00
1586 06AB0 479           GOC    LSTEXT
1587               *
```

```
1588                *
1589                * D1 AT FILE HEADER
1590                *
1591                *
1592 06AB3 7904   LIST50 GOSUB  BASKEY
1593 06AB7 53A           GONC   LSTPOL          NOT BASIC OR KEY FILE TYPE?
1594 06ABA 79C0          GOSUB  PARMXQ          SET UP LINE OR KEY# PARMS IN R1,R3
1595 06ABE 876    LIST60 ?ST=1  6               KEY FILE?
1596 06AC1 4A            GOYES  LSTXKY
1597 06AC3 7431          GOSUB  GETPR1
1598 06AC7 4F8           GOC    LSTER          CARRY=>PRIVATE
1599                *
1600 06ACA 17F    LIST65 D1=D1+ (oFLENh)-(oFTYPh)
1601                * D1 AT FILE LENGTH
1602 06ACD 143           A=DAT1 A
1603 06AD0 8E00          GOSUBL =GETST1
             00
1604                * File end in D(A), DO 2 nibs prior to tEOL
1605 06AD6 DB            C=D    A
1606 06AD8 1F68          D1=(5) =S-R1-1
             8F2
1607 06ADF 145           DAT1=C A               SAVE PTR TO FILE END
1608 06AE2 7C70          GOSUB  LNARGS
1609 06AE6 5F0           GONC   LIST80          ONLY BIGGER LINE# FOUND ?
1610                * D1 AT 1ST LINE TO PRINT
1611 06AE9 8E00   LIST75 GOSUBL =LDCM10         DECOMPILE LINE
             00
1612                * R0 contains pointer past OF
1613 06AEF 7420          GOSUB  PRPSND          Output line & restore endline ptr
1614                *
1615                *
1616                * DONE SENDING LINE - SEE IF THE LIST COMMAND IS FINISHED
1617                *
1618                * MAKE SURE NOT AT END OF PROGRAM
1619                * ENSURE LINE WITHIN RANGE
1620 06AF3 135             D1=C
1621 06AF6 D0     LIST80 A=0    A
1622 06AF8 15B3          A=DAT1 4               READ IN LINE#
1623 06AFC 11B           C=R3                   LAST LINE TO DISPLAY
1624 06AFF 8B6           ?A>C   A               DONE LISTING?
1625 06B02 11            GOYES  LSTDON
1626                *
1627                * BEFORE DECOMPILING NEXT LINE, CHECK FOR ATTN KEY DOWN
1628                * & ANY SERVICE REQUESTS
1629                *
1630 06B04 137           CD1EX                  SAVE D1
1631 06B07 8E00          GOSUBL =CK"ON"
             00
1632 06B0D 135           D1=C
1633 06B10 48D           GOC    LIST75          ATTN key not down?
1634                *
1635 06B13 667E   LSTDON GOTO   nxtstm
1636                *
1637                *
1638                ****************************************************************
```

```
1639            ****************************************************************
1640            **
1641            ** Name:(S) PRPSND  -  Prepare to send buffer to display
1642            **
1643            ** Category:   DSPUTL
1644            **
1645            ** Purpose:
1646            **      Sends buffer to ascii to display device
1647            **
1648            ** Entry:
1649            **      P      =  0
1650            **      HEXMODE
1651            **      B(A)   =  # of characters in buffer
1652            **      OUTBS  =  pointer to start of buffer
1653            **      R0     =  pointer past end of line
1654            **      S-R1-1    contains pointer to end of file
1655            **
1656            ** Exit:
1657            **      P      =  0
1658            **      buffer sent to display
1659            **      C(W)   =  R0
1660            **
1661            ** Calls:      SENDWD, SENDEL, CKINFO
1662            **
1663            ** Uses.......
1664            **  Inclusive: A,B,C,D,D1,D0,R1,R2
1665            **
1666            ** Stk lvls:   5
1667            **
1668            ** NOTE:
1669            **      This routine's integrity requires that for sending a
1670            **      buffer to a display device, SENDWD,SENDEL,CKINFO
1671            **      do not touch R0,R3!!!
1672            **
1673            ** History:
1674            **
1675            **    Date     Programmer             Modification
1676            **  --------   ----------   ----------------------------------
1677            **  10/14/82   S.W.         Wrote routine
1678            **
1679            ****************************************************************
1680            ****************************************************************
1681            *
1682            *
1683 06B17 D9   =PRPSND C=B    A              Save #chars
1684 06B19 134         D0=C
1685 06B1C 8F24        GOSBVL =CKINFO
           581
1686 06B23 136         CDOEX
1687 06B26 D5          B=C    A              Restore #chars
1688 06B28 1FF8        D1=(5) =OUTBS
           5F2
1689 06B2F 147         C=DAT1 A
1690 06B32 854         ST=1   4
1691 06B35 8FF1        GOSBVL =SNDWD+        Send buffer to display device
```

```
            E71
  1692 06B3C 8F1C         GOSBVL =SENDEL        Send CRLF to display device
            D71
  1693 06B43 118          C=R0                  Restore C
  1694 06B46 1B68         D0=(5) =S-R1-1
            8F2
  1695 06B4D 142          A=DAT0 A
  1696 06B50 8BA          ?C>=A  A
  1697 06B53 0C           GOYES  LSTDON
  1698 06B55 01           RTN
  1699              *
  1700              *
  1701              ************************************************************
  1702              ************************************************************
  1703              **
  1704              ** Name:    LNARGS  -  Evaluates Line Number Arguments
  1705              **
  1706              ** Category:   FILUTL
  1707              **
  1708              ** Purpose:    Looks at line number in R1  (assumes PARMXQ has
  1709              **             already been called).  Goes to NXTSTM if no LINE#
  1710              **             found in the specified range.
  1711              **
  1712              ** Entry:      D0 points to initial tEOL of file
  1713              **             D(A) at file end
  1714              **             S10 set if only 1 parm given (set in PARMXQ)
  1715              **             P= 0
  1716              **
  1717              ** Exit:       CARRY SET => D1 POINTS TO LINE MATCHING 1ST PARM
  1718              **                   CLR => D1 POINTS TO BIGGER LINE#
  1719              **                          (LINE# RANGE REQUESTED)
  1720              **             ELSE GOES TO NXTSTM
  1721              **
  1722              ** Calls:      FINDL
  1723              **
  1724              ** Uses:       A,B(A),C(A),D1,D0,S0,S1
  1725              **
  1726              ** Stack lvls: 2
  1727              **
  1728              ** History:
  1729              **
  1730              **    Date     Programmer   Modification
  1731              **  --------   ----------   ------------
  1732              **  06/28/82   S.W.         Added documentation
  1733              **
  1734              ************************************************************
  1735              ************************************************************
  1736              *
  1737 06B57 D8     findlO B=A    A
  1738 06B59 D0            A=0    A
  1739 06B5B 8D00          GOVLNG =FINDLO
            000
  1740              *
  1741 06B62 161    LNARGS D0=D0+ lEOL          Step over initial tEOL
  1742 06B65 136           CDOEX
```

```
1743 06B68 135          D1=C                    D1 at 1st line# of file
1744 06B6B 111          A=R1
1745 06B6E 75EF         GOSUB  findlO
1746 06B72 400          RTNC                    LINE# FOUND?
1747            *
1748 06B75 870          ?ST=1  0                NO LINE# BIGGER?
1749 06B78 B9           GOYES  LSTDON
1750            *
1751 06B7A 87A          ?ST=1  10               ONLY 1 PARM?
1752 06B7D 69           GOYES  LSTDON
1753            *
1754 06B7F 03           RTNCC
1755            *
1756 06B81 8C00  poll   GOLONG =POllj
         00
1757            *
1758            *
1759            *
1760            ********************************************************************
1761            ********************************************************************
1762            **
1763            ** Name:    PARMXQ  -  Checks Line# Range for Validity
1764            **
1765            ** Category:   FILUTL
1766            **
1767            ** Purpose:    'EXECUTES' LINE NUMBER RANGE & PUTS THE SPECIFIED
1768            **             LINE NUMBERS IN R1, R3
1769            **
1770            ** Entry:      DO POINTS AT ALLEGED 1ST PARM
1771            **             P= 0
1772            **             2 ENTRY POINTS:
1773            **             1) PARMXQ - THIS IS THE ENTRY POINT FOR LINE
1774            **                         NUMBER RANGES (DEFAULT IS 1 & 9999) -
1775            **                         WILL CLEAR S9 SO THAT THE RANGE GIVEN
1776            **                         WILL BE VERIFIED AS VALID.
1777            **             2) PRMXQO - USED BY AUTO. S9 SHOULD BE SET SO THE
1778            **                         PARMS GIVEN WILL NOT HAVE VALID RANGE
1779            **                         VERIFIED. Default value in R3
1780            **                         (increment) already set.
1781            **
1782            ** Exit:       R1,R3 contain LINE NUMBERS (S9=0 on entry means
1783            **              default is 1,9999 respectively)
1784            **             S10=1 => FOUND EXACTLY 1 PARM
1785            **                      R3 contains zeroes if S9=0 on entry
1786            **           ERROR EXITS IF 1ST PARM > 2ND PARM
1787            **                      (occurs only IF S9=0 on entry)
1788            **
1789            ** Calls:      none
1790            **
1791            ** Stack lvls: 0
1792            **
1793            ** Uses:       A(A), C, DO, R1, R3, S9, S10
1794            **
1795            ** Detail:     Used by DELETE, MERGE, LIST, AUTO.
1796            **             Assumes that tokenized form of line numbers
```

```
1797                **              has a tCOMMA preceding each line number.
1798                **
1799                ** History:
1800                **
1801                **    Date      Programmer   Modification
1802                **  --------    ----------   ------------
1803                **  06/28/82    S.W.         Added documentation
1804                **
1805                ****************************************************************
1806                ****************************************************************
1807                *
1808 06B87 849     PARMXQ ST=0   9              WANT RANGE CHECKED
1809 06B8A D2             C=0    A
1810 06B8C E6             C=C+1  A
1811 06B8E 109            R1=C                 Parm 1 default = 00001
1812 06B91 3399           LCHEX  9999
          99
1813 06B97 10B            R3=C                 Parm 2 default = 09999
1814                *
1815 06B9A 84A            ST=0   10            1 PARM FLAG
1816 06B9D 14A     PRMXQ0 A=DAT0 B
1817 06BA0 3100           LC(2)  =tCOMMA
1818 06BA4 966            ?A#C   B             NO PARMS?
1819 06BA7 00             RTNYES
1820 06BA9 160            D0=D0+ 1             Advance 1 prior to parm
1821 06BAC 142            A=DAT0 A             Read 1 more than necessary
1822 06BAF F4             ASR    A
1823 06BB1 164            D0=D0+ 5
1824 06BB4 101            R1=A                 WRITE OVER DEFAULT PARM
1825                * JUST IN CASE EXACTLY 1 PARM
1826 06BB7 879            ?ST=1  9             AUTO doesn't want 2nd
1827 06BBA A0             GOYES  PRMXQ1         default parm=0
1828 06BBC 85A            ST=1   10            SET 1-PARM FLAG
1829 06BBF D0             A=0    A
1830 06BC1 103            R3=A                 Set default for parm 2
1831                *
1832 06BC4 14A     PRMXQ1 A=DAT0 B
1833 06BC7 966            ?A#C   B             No 2nd parm?
1834 06BCA 00             RTNYES
1835 06BCC 84A            ST=0   10            CLEAR 1 parm flag
1836 06BCF 161            D0=D0+ 2             Step over tCOMMA
1837 06BD2 15A3           A=DAT0 4
1838 06BD6 103            R3=A                 Read in parm 2
1839 06BD9 879            ?ST=1  9             CALLED BY AUTO?
1840 06BDC 00             RTNYES               Yes=>don't ensure parm1<=parm2
1841 06BDE 119            C=R1
1842 06BE1 8B2            ?C>A   A             parm1>parm2 => error
1843 06BE4 40             GOYES  invarg        ILLEGAL RANGE ERROR
1844 06BE6 01             RTN
1845                *
1846 06BE8 8C00    invarg GOLONG =ARGERR
          00
1847                *
1848                ****************************************************************
1849                ****************************************************************
```

```
1850            **
1851            ** Name:(S) GETPRO  -  Get File Protection of Current File
1852            ** Name:    GETPR+  -  Get File Protection of Specified File
1853            ** Name:    GETPR   -  Get File Protection of Specified File
1854            ** Name:(S) GETPR1  -  Get File Protection of Specified File
1855            **
1856            ** Category:   FILUTL
1857            **
1858            ** Purpose:    Returns file protection information
1859            **
1860            **             GETPRO reads file protection of the current
1861            **             file.
1862            **
1863            **             All other entry points read the file proection
1864            **             nibble of the file specified by the caller.
1865            **
1866            ** Entry:      4 ENTRY POINTS:
1867            **             P= 0
1868            **             1) GETPRO - (CURRST) is accurate
1869            **             2) GETPR+ - (D1) = pointer to file header.
1870            **             3) GETPR  - A(A) = pointer to file header.
1871            **             4) GETPR1 - D1   = pointer to file header.
1872            **
1873            ** Exit:       SB=1 . . . IFF SECURE
1874            **             CARRY SET  IFF PRIVATE
1875            **             D1 POINTS AT FILE TYPE FIELD
1876            **             P=0
1877            **             C(3-0)= eFPROT
1878            **
1879            ** Calls:      none
1880            **
1881            ** Uses:
1882            **      exclusive...  C, D1, SB
1883            **      inclusive...  A(A), C, D1, SB  (GETPRO, GETPR+ only)
1884            **
1885            ** Stack lvls: 0
1886            **
1887            ** History:
1888            **
1889            **    Date     Programmer   Modification
1890            **  --------   ----------   ------------
1891            **  06/28/82   S.W.         Added Documentation
1892            **  10/13/82   S.W.         C(B)=eFPROT on exit
1893            **  11/23/82   S.W.         C(3-0) as above
1894            **
1895            ********************************************************************
1896            ********************************************************************
1897            *
1898            *
1899 06BEE 1FD5 =GETPRO D1=(5) =CURRST
          5F2
1900 06BF5 143  =GETPR+ A=DAT1 A
1901 06BF8 131  =GETPR  D1=A
1902            *
1903 06BFB 17F  =GETPR1 D1=D1+ oFTYPh
```

```
1904 06BFE 173   GETPR2 D1=D1+ lFTYPh          PT TO FLAG FIELD
1905 06C01 14F          C=DAT1 B               Read in prot nib + 1 other
1906 06C04 1C3          D1=D1- lFTYPh
1907 06C07 822          SB=0
1908 06C0A 81E          CSRB
1909             * If SECURE, SB will be set
1910 06C0D 0B           CSTEX
1911 06C0F 870          ?ST=1  0               PRIVATE?
1912 06C12 20           GOYES  GETPR3          Set carry if PRIVATE
1913 06C14 0B    GETPR3 CSTEX
1914 06C16 3300         LC(4)  =eFPROT
          00
1915 06C1C 01           RTN
1916             *
1917             *
1918             ****************************************************************
1919             ****************************************************************
1920             **
1921             ** Name:(S) LSTLEN  -  Calculate #chars to list in display buf
1922             **
1923             ** Category:   DSPUTL
1924             **
1925             ** Purpose:    Calculates number of chars in (display) buffer.
1926             **
1927             ** Entry:      (OUTBS) = Address of buffer start
1928             **             D0      = Address past last character in buffer
1929             **           2 ENTRY POINTS:
1930             **             1) LSTLN+ - 1st calls OUTBYT; preserves 1st
1931             **                         5 nibbles of R0.
1932             **             2) LSTLEN - Ptr to save in C(A)
1933             **
1934             ** Exit:     B(A) = number of characters in buffer
1935             **           Carry clear
1936             **           Pointer saved on entry is restored into R0
1937             **            via OBCOLL (collapse of OUTPUT buffer)
1938             **
1939             ** Calls:    OUTBYT, AVS=D0, OBLCMP, MFWRNQ
1940             **
1941             ** Uses:
1942             **           exclusive... A(A), B, C(A), R0
1943             **           inclusive... A-D, P, D1,D0, R0
1944             **
1945             ** Stack lvls: 5
1946             **
1947             ** Detail:     If #chars to output >=95, then 95 returned
1948             **             as number of characters in buffer and a
1949             **             "Line Too Long" warning is sent out.
1950             **
1951             ** History:
1952             **
1953             **    Date     Programmer   Modifications
1954             **  --------   ----------   -------------
1955             **  07/06/82   S.W.         Improved documentation
1956             **  12/21/82   S.W.         Added 'Line too long'
1957             **
```

```
1958                ****************************************************************
1959                ****************************************************************
1960                *
1961                *
1962 06C1E 8E00 =LSTLN+ GOSUBL =OUTBYT
           00
1963 06C24 118          C=R0                    Save R0 on stack - may be
1964 06C27 06   =LSTLEN RSTK=C                   destroyed by MFWRNQ
1965 06C29 8E00         GOSUBL =AVS=D0          Set AVMEMS to D0
           00
1966 06C2F 8E00         GOSUBL =OBLCMP          Calculate length of buffer
           00
1967                * Length in A(A) [in nibbles]
1968 06C35 D2           C=0    A
1969 06C37 31EB         LC(2)  2*95             95 character maximum
1970 06C3B 8BA          ?A<=C  A
1971 06C3E 61           GOYES  LSTLOK
1972                * Give warning 'line too long'
1973 06C40 3300         LC(4)  =eL2LNG
           00
1974 06C46 8E00         GOSUBL =MFWRQ8          Quiet option
           00
1975                *
1976 06C4C D2           C=0    A
1977 06C4E 31EB         LC(2)  2*95
1978 06C52 DA           A=C    A                Set #chars to 95
1979                *
1980 06C54 AF1   LSTLOK B=0    W
1981 06C57 D8           B=A    A
1982 06C59 81D          BSRB                    #CHARACTERS IN BUFFER
1983 06C5C 07           C=RSTK
1984 06C5E 108          R0=C
1985 06C61 8C00  obcoll GOLONG =OBCOLL          Destroys D1,C(A)
           00
1986                *
1987                *
1988                ****************************************************************
1989                ****************************************************************
1990                **
1991                ** Name:    MERGE   -  Executes MERGE Command
1992                **
1993                ** Category:   STEXEC
1994                **
1995                ** Purpose:    MERGES BASIC FILE IN WITH CURRENT FILE.
1996                **             MERGES KEY FILE IN WITH keys FILE.
1997                **
1998                ** Entry:      DO PAST tMERGE
1999                **
2000                ** Exit:       via NXTSTM
2001                **
2002                ** Calls:      FINDF, NXTLIN, PEDIT, POLL, GETPRO, FSPECx,
2003                **             KEYFND, KEYMRG, BASKEY, FILBLK, GETST*, PSHUPD
2004                **             KYARGS, KMEMCK, CHKPSF, ZERPGM, PARMXQ, DO=GSB,
2005                **             CURDVC, FLADDR, MRGUPD, AVS=C, MOVEUM, POPSTK,
2006                **             RMEMCH
```

```
2007            **
2008            ** Uses:       S0,S1,S6,S7,S8,S9,S10
2009            **             F-R0-1 (KEYMRG), S-R0-0,S-R0-1
2010            **
2011            **             A-D, D1,D0, R0-R3, S1,S2,S7, STMTD0
2012            **             STMTR1 (All of it), All of function scratch
2013            **
2014            ** Detail:     PROGRAMMABLE, however:
2015            **             Attempting to MERGE a BASIC file during a running
2016            **             program results in the MERGE being executed, then
2017            **             an ENDALL taking place. (Any following statements
2018            **             will not be executed).  This INCLUDES the case of
2019            **             MERGE of a BASIC file that really doesn't alter
2020            **             current file, as in the case of an empty source
2021            **             file.
2022            **
2023            ** History:
2024            **
2025            **    Date     Programmer   Modification
2026            **  --------   ----------   ----------------------------
2027            **  06/28/82   S.W.         Added Documentation
2028            **  10/15/82   S.W.         Calls LAKEYS before searching
2029            **                           for keys file
2030            **  03/03/83   J.P.         CHKPSF call before PEDITM
2031            **
2032            ********************************************************************
2033            ********************************************************************
2034            *
2035            *
2036            ********************************************************************
2037            ********************************************************************
2038            **
2039            ** Name:(S) pMERGE  -  Polls to MERGE non-mainframe file
2040            **
2041            ** Category:   POLL
2042            **
2043            ** Type:       POLL
2044            **
2045            ** Purpose:
2046            **      Polls to MERGE a non-mainframe file
2047            **
2048            ** Should poll be "Handled" (return with XM=0)?:
2049            **      Yes
2050            **
2051            ** Meaning of "Handling" Poll (what does code do if handled?):
2052            **      Merges designated file into current file (if BASIC),
2053            **      into keys file (if KEY), or other if some other file
2054            **      type and the command has been extended to allow this.
2055            **
2056            ** Entry conditions for handler (registers, ST, RAM, etc.):
2057            **      Carry clear
2058            **      B[A] = Poll number.
2059            **      HEX mode.
2060            **      P=0.
2061            **      A(W) contains first 8 characters of file name
```

```
2062            **      R0(3-0) contains characters 9 & 10
2063            **      D0 past file specifier
2064            **
2065            ** Normal exit conditions from handler if handled (ST, RAM,
2066            ** registers, etc.):
2067            **      Carry clear
2068            **      HEX mode.
2069            **      XM=0.
2070            **      PCADDR intact, ready to go on to NXTSTM.
2071            **
2072            ** Normal exit conditions from handler if not handled (ST, RAM,
2073            ** registers, etc.):
2074            **      Carry clear
2075            **      HEX mode.
2076            **      XM=1.
2077            **      R0 MUST be preserved from entry.
2078            **
2079            ** Error exit conditions from handler:
2080            **      Carry set.
2081            **      HEX mode.
2082            **      C[0-3] = Error number.
2083            **
2084            ** Available subroutine levels:
2085            **      7
2086            **
2087            ** NOTE:
2088            **      For no file name specified (MERGE :<device>)
2089            **      the default error message for 'not handled' will
2090            **      be eFSPEC.
2091            **
2092            ** What registers/RAM may be used if handled?:
2093            **      A-D, D0, D1, P
2094            **      R0-R4, all statuses except S13
2095            **      All of STMT and FN scratch
2096            **
2097            ** What registers/RAM may be used if not handled?:
2098            **      A-D, D0, D1, P
2099            **      R1,R2,R3; All statuses except S13
2100            **      R0 can NOT be altered!
2101            **      All of STMT and FN scratch
2102            **
2103            ** What registers/RAM may be used if error exit (POLL only)?:
2104            **      A-D, D0, D1, P
2105            **      R0-R4, All statuses except S13
2106            **      All of STMT and FN scratch
2107            **
2108            ** Envisioned application(s):
2109            **      Note that poll handler must check the following:
2110            **      1) file type of specified file
2111            **      2) Protection of source (can't be PRIVATE), and
2112            **         of destination (can't be SECURE or PRIVATE).
2113            **      3) Destination must be in RAM
2114            **      4) Sufficient memory?
2115            **
2116            ** History:
```

```
2117            **
2118            **    Date     Programmer          Modification
2119            **  --------   ----------   ----------------------------------
2120            **  04/18/83   S.W.         Updated documentation
2121            **
2122            ********************************************************************
2123            ********************************************************************
2124            *
2125            ********************************************************************
2126            ********************************************************************
2127            **
2128            ** Name:(S) pMRGE2  -  Polls to MERGE non-BASIC,non-KEY file
2129            **
2130            ** Category:   POLL
2131            **
2132            ** Type:       POLL
2133            **
2134            ** Purpose:
2135            **      Polls for handling of MERGE of non-BASIC,non-KEY
2136            **
2137            ** Should poll be "Handled" (return with XM=0)?:
2138            **      Yes
2139            **
2140            ** Meaning of "Handling" Poll (what does code do if handled?):
2141            **      Does appropriate MERGE, checking file protection, and
2142            **      memory requirements, exits ready to go on to NXTSTM.
2143            **
2144            ** Entry conditions for handler (registers, ST, RAM, etc.):
2145            **      Carry clear
2146            **      B[A] = Poll number.
2147            **      HEX mode.
2148            **      P=0.
2149            **      D1 at start of mainframe (source) file header
2150            **      A(A)=File type#
2151            **      D0 past file specifier
2152            **
2153            ** Normal exit conditions from handler if handled (ST, RAM,
2154            ** registers, etc.):
2155            **      Carry clear.
2156            **      HEX mode.
2157            **      XM=0.
2158            **      RFADJ has been called to update necessary pointers,etc
2159            **      Ready to go on to NXTSTM.
2160            **
2161            ** Normal exit conditions from handler if not handled (ST, RAM,
2162            ** registers, etc.):
2163            **      Carry clear
2164            **      HEX mode.
2165            **      XM=1.
2166            **
2167            ** Error exit conditions from handler (POLL only):
2168            **      Carry set.
2169            **      HEX mode.
2170            **      C[0-3] = Error number.
2171            **
```

```
2172                ** Available subroutine levels:
2173                **      7
2174                **
2175                ** NOTE:
2176                **
2177                **      --STMT/FN Scratch, SCRTCH, SNAPBF, TRFMBF, LDCSPC,--
2178                **      --LEXPTR.--
2179                **
2180                ** What registers/RAM may be used if handled?:
2181                **      A-D, D0, D1, P
2182                **      R0-R4, All statuses except S13
2183                **      All of STMT and FN scratch.
2184                **
2185                ** What registers/RAM may be used if not handled?:
2186                **      A-D, D0, D1, P
2187                **      R0-R4, All statuses except S13
2188                **      All of STMT and FN scratch.
2189                **
2190                ** What registers/RAM may be used if error exit (POLL only)?:
2191                **      A-D, D0, D1, P
2192                **      R0-R4, All statuses except S13
2193                **      All of STMT and FN scratch.
2194                **
2195                ** Envisioned application(s):
2196                **      Perhaps merging TEXT or LEX files.
2197                **      Could implement by using the EDIT poll to position
2198                **      at the file, thereby making it the current file.
2199                **
2200                ** History:
2201                **
2202                **    Date     Programmer          Modification
2203                **  --------   ----------   ----------------------------------
2204                **  04/18/83   S.W.         Added documentation
2205                **
2206                ************************************************************
2207                ************************************************************
2208                *
2209                *
2210                *
2211                * TRY MERGE ON NON-MAINFRAME FILE
2212 06C67 876       MRGEXT ?ST=1  6
2213 06C6A 42                GOYES  MRGERR
2214 06C6C 711F              GOSUB  poll
2215 06C70 D0                CON(2) =pMERGE
2216 06C72 6E59              GOTO   CAT62          DEFAULT HANDLER - ILL FILE SPEC
2217                *
2218                * TRY MERGE ON FILE TYPE OTHER THAN BASIC OR KEY
2219 06C76 07     MRGPOL C=RSTK                  PULL LOCATION INFO OFF STACK
2220 06C78 D7                D=C    A
2221 06C7A 730F              GOSUB  poll
2222 06C7E F2                CON(2) =pMRGE2
2223 06C80 4D0  =pPOLL3 GOC    MRGERR          HANDLED & ERRORED?
2224                *
2225 06C83 831               ?XM=0                  HANDLED & OKAY?
2226 06C86 21                GOYES  MRGFIN
```

```
2227            * DEFAULT HANDLER - ILL. FILE TYPE
2228 06C88 3300          LC(4)  =eFTYPE
           00
2229            *
2230 06C8E 6059  MRGERR GOTO   bserr
2231            *
2232 06C92 8C00 =FSPECj GOLONG =FSPECx
           00
2233            *
2234 06C98 61FC  MRGFIN GOTO   nxtstm
2235            *
2236 06C9C 0000          REL(5) =MERGDC
           0
2237 06CA1 0000          REL(5) =MERGEP
           0
2238 06CA6 78EF =MERGE   GOSUB  FSPECj
2239 06CAA 43E           GOC    MRGERR
2240            * LEGAL FILE SPECIFIER - SEE IF FILE EXISTS
2241 06CAD 8E00          GOSUBL =FINDF+
           00
2242 06CB3 43B           GOC    MRGEXT
2243 06CB6 133           AD1EX
2244 06CB9 131           D1=A
2245 06CBC 100           R0=A                  Save ptr to file header (keys)
2246            * ENSURE IT'S A FILE OF TYPE BASIC OR KEY
2247 06CBF 7DF1          GOSUB  BASKEY
2248 06CC3 52B           GONC   MRGPOL         NOT BASIC OR KEY FILE?
2249 06CC6 713F          GOSUB  GETPR1
2250 06CCA 43C   MERG04  GOC    MRGERR         Error if PRIVATE
2251            *
2252 06CCD 76BE  MERG06  GOSUB  PARMXQ         Set up line#/key# parms in R1,R3
2253 06CD1 110           A=R0
2254            *
2255 06CD4 866           ?ST=0  6              BASIC FILE MERGE?
2256 06CD7 03            GOYES  MERG30
2257 06CD9 6EC0          GOTO   MERG62
2258            *
2259 06CDD 1B3A  D0=GSB  D0=(5) =GSBSTK
           5F2
2260 06CE4 142           A=DAT0 A
2261 06CE7 130           D0=A
2262 06CEA 166           D0=D0+ 7
2263 06CED 142           A=DAT0 A
2264 06CF0 01            RTN
2265            *
2266 06CF2 06    PSHADR  RSTK=C                Save Block End
2267 06CF4 D9            C=B    A
2268 06CF6 10B           R3=C                  Save Block Size
2269 06CF9 7400          GOSUB  pshupd         Push Block Start on Stk
2270 06CFD 07            C=RSTK
2271 06CFF DA            A=C    A
2272 06D01 8C00  pshupd  GOLONG =PSHUPD        Push block end on stk
           00
2273
2274            *
```

```
2275 06D07 8E00  MERG30 GOSUBL =GETST*
           00
2276            * D at end of file
2277 06D0D 7581         GOSUB  FILBLK
2278 06D11 7DDF         GOSUB  PSHADR
2279            *
2280 06D15 8E00         GOSUBL =CURDVC        Set-up for FLADDR
           00
2281 06D1B 8E00         GOSUBL =FLADDR
           00
2282            *
2283 06D21 12B          CR3EX
2284 06D24 D5           B=C    A              Amt of mem needed
2285 06D26 11B          C=R3                  Restore C(A)
2286 06D29 8E00         GOSUBL =RMEMCH
           00
2287 06D2F 4A9          GOC    MERG04
2288            * Error if PRIVATE, SECURE, or non-BASIC
2289 06D32 8E00         GOSUBL =CHKPSF
           00
2290            *
2291            * COMPLETE SET-UP FOR 'PEDIT' CALL
2292            *
2293 06D38 7EF0  MERG50 GOSUB  MRGUPD
2294            *
2295 06D3C 4B3          GOC    MERG59
2296 06D3F 8F00         GOSBVL =NXTLIN
           000
2297            * C(A) contains end source
2298 06D46 1FF8         D1=(5) =OUTBS
           5F2
2299 06D4D 143          A=DAT1 A
2300 06D50 131          D1=A                  Begin Dest.
2301            *
2302 06D53 768F         GOSUB  D0=GSB
2303 06D57 144          DAT0=C A              UPDATE
2304            * Move line to output buffer
2305 06D5A 8F00         GOSBVL =MOVEU2        Move line to output buffer
           000
2306            *
2307 06D61 137          CD1EX
2308 06D64 8E00         GOSUBL =AVS=C         Update AVMEMS
           00
2309 06D6A 848          ST=0   8              Flags that this isn't DELETE
2310 06D6D 8F00         GOSBVL =PEDITM
           000
2311 06D74 63CF         GOTO   MERG50
2312            *
2313            * S13 here should indicate whether the MERGE is executed from
2314            * program execution or from keyboard BASIC
2315            *
2316 06D78 86D   MERG59 ?ST=0  13             CALC BASIC?
2317 06D7B 80           GOYES  MERG60         IF SO, GOTO MERG60
2318 06D7D 8C00         GOLONG =ENDALL        GOSUB stack will be collapsed
           00
```

```
2319                *
2320 06D83 8E00     MERG60 GOSUBL =ZERPGM        Clear prog info (timers,etc)
           00
2321 06D89 1B3A     MERG61 DO=(5) =GSBSTK
           5F2
2322                * A points to start of entry to delete
2323 06D90 142             A=DAT0 A
2324 06D93 D2              C=0    A
2325 06D95 30C             LCHEX  C
2326 06D98 C2              C=C+A  A              C pts to entry end
2327 06D9A DE              ACEX   A
2328 06D9C 22              P=     2              Update 3 pointers
2329 06D9E 8E00            GOSUBL =POPSTK
           00
2330 06DA4 65EB            GOTO   nxtstm
2331                *
2332                * NO NEED TO CHECK PRIVACY
2333                *
2334 06DA8 131      MERG62 D1=A
2335 06DAB 8F00            GOSBVL =KYARGS
           000
2336 06DB2 7890            GOSUB  PRMSV          SET UP S-R0-0 & S-R0-1
2337 06DB6 5F1             GONC   MERG67         SINGLE PARM SPECIFIED?
2338                *  KEY# RANGE
2339 06DB9 120             AROEX                 Rest file ptr (save blk st)
2340 06DBC 11B             C=R3
2341 06DBF D7              D=C    A              KEYCODE
2342 06DC1 8E00            GOSUBL =KYFND+
           00
2343 06DC7 133             AD1EX
2344 06DCA 540             GONC   MERG65         MATCH NOT FOUND?
2345                * MATCH FOUND - PT TO END OF KEY ASSIGNMENT
2346 06DCD CA              A=A+C  A
2347                *
2348 06DCF 118      MERG65 C=R0
2349                * Block start in C(A); Block end in A(A)
2350 06DD2 72E0            GOSUB  PRMSV7         CALCULATE MEMORY NEEDED
2351                *
2352                * Store (Block end - 1) so that pointer doesn't inadvertantly
2353                * fall on file boundary (eg at end of file chain)
2354 06DD6 CE       MERG67 C=C-1  A
2355 06DD8 761F            GOSUB  PSHADR
2356 06DDC 11B             C=R3
2357 06DDF D5              B=C    A
2358 06DE1 8E00            GOSUBL =KMEMCK
           00
2359 06DE7 72FE            GOSUB  DO=GSB
2360 06DEB D6              C=A    A
2361 06DED 5E1             GONC   MERG69         (B.E.T.)
2362                *
2363                * MERGE FILE INTO keys
2364                *
2365 06DF0 7640     MRGKEY GOSUB  MRGUPD
2366                *
2367                * INCREMENT RAM PTR TO NEXT KEY ASSIGNMENT
```

```
2368 06DF4 D5          B=C    A              End of block FROM 'MRGUPD'
2369 06DF6 D2          C=0    A
2370 06DF8 171         D1=D1+ 2
2371 06DFB 14F         C=DAT1 B
2372 06DFE C2          C=A+C  A              PT TO NEXT KEY ASSIGN.
2373             * EOF NOT CAUGHT BY MRGUPD, SINCE DON'T STORE NEXT KEY TO
2374             * MERGE, BUT ONLY THE CURRENT ONE - DIFFERENT FROM THE WAY
2375             * BASIC MERGE STORES IT
2376 06E00 8BD         ?C>=B  A              AT EOF?
2377 06E03 68          GOYES  MERG61
2378             *
2379 06E05 74DE        GOSUB  DO=GSB
2380 06E09 144         DAT0=C A              UPDATE PTR
2381 06E0C 135   MERG69 D1=C
2382 06E0F AF2         C=0    W
2383 06E12 15F3        C=DAT1 4
2384 06E16 D1          B=0    A
2385 06E18 AE5         B=C    B
2386 06E1B 8E00        GOSUBL =KYMRG+
           00
2387 06E21 78BE        GOSUB  DO=GSB         PTR TO NEXT KEY ASSIGN TO MERGE
2388             * BGN DEST IN D1;BGN SOURCE IN A(A)
2389 06E25 130         DO=A                  Bgn source
2390 06E28 161         DO=DO+ 2
2391 06E2B D2          C=0    A
2392 06E2D 14E         C=DAT0 B
2393 06E30 8F00        GOSBVL =MOVEU4
           000
2394             *
2395 06E37 58B         GONC   MRGKEY         (B.E.T.)
2396             *
2397             *
2398             ************************************************************
2399             ************************************************************
2400             **
2401             ** Name:     MRGUPD  -  merge update
2402             **
2403             ** Category:   LOCAL
2404             **
2405             ** Purpose:    Compares END OF BLOCK PTR and CURRENT MERGE LINE
2406             **            PTR to see if merge is complete
2407             **
2408             ** Entry:      B=OFFSET
2409             **            ADDRESS OF NEXT LINE TO MERGE & ADDRESS OF END
2410             **            OF BLOCK TO MERGE PUSHED ON GOSUB STACK
2411             **
2412             ** Exit:       CARRY SET=> DONE MERGING
2413             **            A(A),D1 point at current line
2414             **            C(A) contains pointer to end of block
2415             **
2416             ** Calls:      none
2417             **
2418             ** Uses:       A(A), B(A), C(A), D1, DO
2419             **
2420             ** Detail:     USED BY MERGE ONLY
```

```
2421                    **
2422                    ** History:
2423                    **
2424                    **    Date     Programmer   Modification
2425                    **  --------   ----------   ------------
2426                    **  06/28/82   S.W.         Added documentation
2427                    **
2428                    ********************************************************************
2429                    ********************************************************************
2430                    *
2431                    *
2432 06E3A 7F9E  MRGUPD GOSUB  D0=GSB
2433                    *
2434 06E3E 185          D0=D0- 6
2435 06E41 146          C=DAT0 A
2436 06E44 131          D1=A
2437 06E47 8BE          ?A>=C  A                DONE?
2438 06E4A 00           RTNYES
2439                    *
2440 06E4C 03   MRGUP1  RTNCC
2441                    *
2442 06E4E 133  PRMSV   AD1EX
2443 06E51 131          D1=A
2444 06E54 86A          ?ST=0  10               NOT SINGLE PARM?
2445 06E57 00           RTNYES
2446 06E59 D2           C=0    A
2447 06E5B 876          ?ST=1  6                KEY FILE?
2448 06E5E C2           GOYES  PRMSV2
2449 06E60 173          D1=D1+ 4
2450                    * D1 AT LENGTH BYTE
2451 06E63 14F          C=DAT1 B
2452 06E66 C2           C=A+C  A
2453 06E68 134          D0=C
2454 06E6B 163          D0=D0+ 4                Point to t@ or tEOL
2455 06E6E 3100         LC(2)  =tEOL
2456 06E72 8E00         GOSUBL =EOLSN5
             00
2457 06E78 161          D0=D0+ 2                D0 past block
2458 06E7B 1C3          D1=D1- 4                D1 at block start
2459 06E7E 136          CD0EX                   Block end
2460 06E81 133          AD1EX                   Block start
2461 06E84 D5           B=C    A
2462 06E86 E8           B=B-A  A                Block length
2463 06E88 03           RTNCC
2464                    *
2465 06E8A 171  PRMSV2  D1=D1+ 2
2466 06E8D 14F          C=DAT1 B
2467 06E90 D5           B=C    A                Length of line
2468 06E92 C2           C=A+C  A                End of line
2469 06E94 03           RTNCC
2470                    *
2471                    ********************************************************************
2472                    ********************************************************************
2473                    **
2474                    ** Name:    FILBLK  -  Delineates Specified File Block
```

```
2475                **
2476                ** Category:   FILUTL
2477                **
2478                ** Purpose:
2479                **             Given starting and ending BASIC program line
2480                **             numbers, FILBLK computes the address of the
2481                **             block start and end, and the block size.
2482                **
2483                ** Entry:
2484                **       D(A) points to file end
2485                **       P=0
2486                **       DO points at initial tEOL
2487                **       Exit conditions from PARMXQ:
2488                **       R1 = Start line#
2489                **       R3 = End line#
2490                **
2491                ** Exit:
2492                **       If no program memory is in specified range, FILUTL
2493                **        does a 'GOLONG NXTSTM'
2494                **             ELSE...
2495                **                  A(A) = Block start
2496                **                  C(A) = Block end
2497                **                  B(A) = Block size
2498                **
2499                ** Calls:       LNARGS, FINDL, NXTLIN, PRMSV
2500                **
2501                ** Uses:        A, B(A), C(A), D1,D0
2502                **
2503                ** Stk lvls:    3
2504                **
2505                ********************************************************************
2506                ********************************************************************
2507 06E96 78CC     FILBLK GOSUB  LNARGS
2508 06E9A 70BF            GOSUB  PRMSV          SET UP A, C
2509 06E9E 500             RTNNC                 SINGLE PARM SPECIFIED?
2510 06EA1 113             A=R3
2511                * C(A) POINTS TO START OF BLOCK TO MERGE
2512 06EA4 06              RSTK=C                Save block start
2513 06EA6 7DAC            GOSUB  findl0
2514 06EAA 590             GONC   PRMSV5         MATCH NOT FOUND?
2515                * MATCH FOUND - PT TO END OF LINE
2516 06EAD 8F00            GOSBVL =NXTLIN
           000
2517                * C(A) POINTS AT END OF BLOCK TO MERGE
2518                *
2519 06EB4 DA     PRMSV5 A=C    A               Block end
2520 06EB6 07            C=RSTK                 Block start
2521 06EB8 D8     PRMSV7 B=A    A               Block end
2522 06EBA E1            B=B-C  A               Block length
2523 06EBC DE            ACEX   A               A=Blk start;C=Blk end
2524 06EBE 03            RTNCC
2525                *
2526                ********************************************************************
2527                ********************************************************************
2528                **
```

```
2529            ** Name:    BASKEY  -  Determine if File Type is BASIC/KEY
2530            **
2531            ** Category:   FILUTL
2532            **
2533            ** Purpose:    Checks if indicated file is of type BASIC or KEY
2534            **
2535            ** Entry:      D1 = Address of file header start
2536            **
2537            ** Exit:       D1 = Address of file header start
2538            **             P= 0
2539            **             A(A) = file type#
2540            **             S6=0 => BASIC file
2541            **             CARRY SET => BASIC or KEY file
2542            **                          S6=0 => BASIC
2543            **                   CLR => Neither BASIC nor KEY file (S6=1)
2544            **
2545            ** Calls:      none
2546            **
2547            ** Uses:       A(A), C(A), S6
2548            **
2549            ** Stk lvls:   0
2550            **
2551            ** History:
2552            **
2553            **    Date     Programmer   Modification
2554            **  --------   ----------   ------------
2555            **  06/28/82   S.W.         Added Documentation
2556            **
2557            ************************************************************
2558            ************************************************************
2559            *
2560            *
2561 06EC0 846  =BASKEY ST=0   6
2562 06EC3 17E         D1=D1+ (oFTYPh)-1
2563            * Read in 1 more than needed
2564 06EC6 143         A=DAT1 A
2565 06EC9 F4          ASR    A              Shift off extra nibble
2566 06ECB 1CE         D1=D1- (oFTYPh)-1
2567 06ECE 3441        LC(5)  =fBASIC        Load zero in nib 4
          2E0
2568 06ED5 8A2         ?A=C   A
2569 06ED8 00          RTNYES
2570            *
2571 06EDA 856         ST=1   6
2572 06EDD 33C0        LC(4)  =fKEY
          2E
2573 06EE3 8A2         ?A=C   A
2574 06EE6 00          RTNYES
2575 06EE8 01          RTN
2576            *
2577            *
2578            ************************************************************
2579            ************************************************************
2580            **
2581            ** Name:    EOFLCH  -  Scan to End of File Chain
```

```
2582                ** Name:    EOFLC+  -  Scan to End of File Chain
2583                ** Name:    EOFLC1  -  Scan to End of File Chain
2584                **
2585                ** Category:   FILUTL
2586                **
2587                ** Purpose:    Scans to end of file chain
2588                **
2589                **            EOFLCH entry requires a pointer to the
2590                **            start of a known file.
2591                **
2592                **            EOFLC+ and EOFLC1 entry require a pointer
2593                **            to the end of some file in the chain.  The
2594                **            caller need not know that a file follows
2595                **            since the code for these entry points checks
2596                **            for a 00 byte instead of assuming a file
2597                **            follows.
2598                **
2599                ** Entry:      P= 0
2600                **            3 entry points:
2601                **            1) EOFLCH - C(A) points to file header
2602                **            2) EOFLC+ - C(A) at end of some file. (Either
2603                **                        at start of next file header or
2604                **                        at 00 byte at end of file chain)
2605                **            3) EOFLC1 - D1,C(A) at END of some file
2606                **
2607                ** Exit:       Carry Clear
2608                **            C(A) & D1 point to chain end  (at 00 byte)
2609                **
2610                ** Calls:      FILSKP
2611                **
2612                ** Uses:       A(A), C(A), D1
2613                **
2614                ** Stack lvls: 1
2615                **
2616                ** History:
2617                **
2618                **    Date      Programmer   Modification
2619                **  ---------  ----------   ------------
2620                **  06/28/82   S.W.         Added Documentation
2621                **  10/21/82   S.W.         No longer uses B
2622                **  01/12/83   S.W.         Eliminated EOFLC* entry
2623                **
2624                ****************************************************************
2625                ****************************************************************
2626                *
2627                *
2628 06EEA 7D20 =EOFLCH GOSUB  FILSKP
2629 06EEE 135  =EOFLC+ D1=C
2630 06EF1 14B  =EOFLC1 A=DAT1 B
2631 06EF4 96C          ?A#0   B
2632 06EF7 3F           GOYES  EOFLCH
2633                *
2634 06EF9 01           RTN
2635                *
2636                *
```

```
2637            ****************************************************************
2638            ****************************************************************
2639            **
2640            ** Name:    CLMPRO  -  CLAIM Protection
2641            **
2642            ** Category:   FILUTL
2643            **
2644            ** Purpose:
2645            **      Checks for secure files in file chain out on Plug-in.
2646            **      Needed if user requests CLAIMPORT operation
2647            **
2648            ** Entry:
2649            **      P      =  0
2650            **      C(A) at 1st file in chain (assumes file chain not null)
2651            **
2652            ** Exit:
2653            **      P      =  0
2654            **      CARRY SET; D1 AT ZERO BYTE AT END OF CHAIN
2655            **      ERROR EXITS IF ANY FILE IN THE CHAIN IS SECURE
2656            **
2657            ** Calls:      GETPRO, FILSKP
2658            **
2659            ** Uses.......
2660            **             A(A), B(A), C, D1, SB
2661            **
2662            ** Stk lvls:   1
2663            **
2664            ** History:
2665            **
2666            **    Date      Programmer              Modification
2667            **  --------    ----------   ---------------------------------
2668            **  06/25/82    S.W.         CREATED UTILITY
2669            **  01/13/83    S.W.         Modified entry conditions
2670            **
2671            ****************************************************************
2672            ****************************************************************
2673            *
2674 06EFB 7A10  CLMPR1 GOSUB  FLSKPB
2675 06EFF 135  =CLMPRO D1=C
2676 06F02 14B         A=DAT1 B
2677 06F05 968         ?A=0   B
2678 06F08 00          RTNYES
2679 06F0A D5          B=C    A              SAVE PTR TO FILE HEADER
2680 06F0C 7BEC        GOSUB  GETPR1
2681 06F10 832         ?SB=0                 FILE NOT SECURE?
2682 06F13 8E          GOYES  CLMPR1
2683 06F15 687D        GOTO   MRGERR
2684            *
2685            *
2686            ****************************************************************
2687            ****************************************************************
2688            **
2689            ** Name:    FILSKP  -  File Skip
2690            ** Name:    FLSKPB  -  File Skip
2691            ** Name:(S) FILSK+  -  File Skip
```

```
2692                **
2693                ** Category:   FILUTL
2694                **
2695                ** Purpose:
2696                **      Skips over specified file
2697                **
2698                ** Entry:
2699                **      P=0
2700                **      3 entry points:
2701                **      1) FLSKPB - B(A) at file header start
2702                **      2) FILSKP - C(A) at file header start
2703                **      3) FILSK+ - A(A) at file header start
2704                **
2705                ** Exit:
2706                **      P=0
2707                **      C(A)= Points to next file in chain (OR to 00 BYTE)
2708                **      A(A)= Length in file's file length field
2709                **      D1  = Points to file length field
2710                **      Carry clear
2711                **
2712                ** Calls:      none
2713                **
2714                ** Uses A(A), C(A), D1
2715                **
2716                ** Stk lvls:   0
2717                **
2718                ** History:
2719                **
2720                **    Date      Programmer   Modifications
2721                **  --------    ----------   -------------
2722                **  07/05/82    S.W.         Added documentation
2723                **  10/21/82    S.W.         Changed entry conditions
2724                **
2725                ******************************************************************
2726                ******************************************************************
2727                *
2728 06F19 D9       =FLSKPB C=B    A
2729 06F1B DA       =FILSKP A=C    A
2730 06F1D D2       =FILSK+ C=0    A
2731 06F1F 3102             LC(2)  =oFLENh
2732 06F23 C2               C=C+A  A
2733 06F25 135              D1=C
2734 06F28 143              A=DAT1 A              WAS C=DAT1 A; AD1EX
2735 06F2B C2               C=A+C  A
2736 06F2D 01               RTN
2737                *
2738                *
2739 06F2F                  END
```

```
 ARGERR  Ext                  -  1846
 AUTINC  Abs  194251 #2F6CB   -    12  1244
 AUTLIN  Abs   27118 #069EE   -  1267  1246
=AUTLN2  Abs   27130 #069FA   -  1275
=AUTO    Abs   27035 #0699B   -  1235
 AUTOP   Ext                  -  1233
 AUTXQ5  Abs   27116 #069EC   -  1265  1262
 AUTXQ6  Abs   27120 #069F0   -  1268  1257
=AUTXQ7  Abs   27081 #069C9   -  1252
 AUTXQ8  Abs   27167 #06A1F   -  1287  1269
 AVM2DS  Ext                  -   448
 AVS=C   Ext                  -  2308
 AVS=D0  Ext                  -   719  1965
=BASKEY  Abs   28352 #06EC0   -  2561  1592  2247
 BF2DSP  Ext                  -   226
 BF2STK  Abs   99939 #18663   -    13   786
 BLNKC+  Ext                  -   707
 BSERR   Ext                  -   359
 C=MAIN  Ext                  -   214   649
=CAT     Abs   25644 #0642C   -   177
=CAT$    Abs   26216 #06668   -   615
 CAT$05  Abs   26309 #066C5   -   651   692
 CAT$08  Abs   26314 #066CA   -   654   662
 CAT$12  Abs   26374 #06706   -   682   646
 CAT$14  Abs   26406 #06726   -   694   368
 CAT$15  Abs   26429 #0673D   -   703   652
=CAT$20  Abs   26438 #06746   -   707   446   660
 CAT$30  Abs   26452 #06754   -   714   717
 CAT$32  Abs   26500 #06784   -   731   729
 CAT$35  Abs   26509 #0678D   -   735   732
=CAT$65  Abs   26622 #067FE   -   777   697   770   772
=CAT$66  Abs   26634 #0680A   -   783
 CAT$70  Abs   26770 #06892   -   940   746   762
 CAT$71  Abs   26794 #068AA   -   950   754   851
 CAT$75  Abs   26804 #068B4   -   953   959
 CAT$80  Abs   26826 #068CA   -  1014   741
=CAT$83  Abs   26836 #068D4   -  1019
=CAT$90  Abs   26839 #068D7   -  1020  1093
 CAT$95  Abs   26860 #068EC   -  1023  1090
 CAT$ER  Abs   26370 #06702   -   678   671
 CAT$N   Abs   26260 #06694   -   634   621
 CAT$NL  Abs   26412 #0672C   -   696   657   704
 CAT$P1  Abs   26339 #066E3   -   664   673
 CAT$PL  Abs   26350 #066EE   -   669   625   629
 CAT*    Abs   25624 #06418   -   172   170
 CAT00   Abs   25657 #06439   -   185   336
 CAT05   Abs   25675 #0644B   -   198   178
 CAT10   Abs   25715 #06473   -   214   205   353
=CAT100  Abs   26200 #06658   -   448
 CAT11   Abs   25721 #06479   -   215   377
 CAT16   Abs   25760 #064A0   -   228   255
 CAT17   Abs   25784 #064B8   -   237   283
 CAT18   Abs   25794 #064C2   -   240   282   294
 CAT19   Abs   25873 #06511   -   269   251
 CAT21   Abs   25909 #06535   -   286   253   321
```

```
 CAT22   Abs   25921 #06541 -   289   304
 CAT25   Abs   25935 #0654F -   296   290
 CAT30   Abs   25961 #06569 -   306   259
 CAT31   Abs   25982 #0657E -   313   264   370
 CAT32   Abs   25999 #0658F -   318   257
 CAT37   Abs   26014 #0659E -   326   171
 CAT38   Abs   26021 #065A5 -   329   211
 CAT40   Abs   26034 #065B2 -   338   330
 CAT62   Abs   26065 #065D1 -   351  1553  2216
 CAT65   Abs   26071 #065D7 -   353   345
 CAT67   Abs   26075 #065DB -   355   341
 CAT70   Abs   26085 #065E5 -   361   347
 CAT80   Abs   26108 #065FC -   370   384   388
 CAT82   Abs   26112 #06600 -   373   362
 CAT83   Abs   26121 #06609 -   376   394
 CAT85   Abs   26150 #06626 -   387   391
 CAT87   Abs   26156 #0662C -   388   398
 CAT90   Abs   26159 #0662F -   389   366
 CAT95   Abs   26189 #0664D -   445   186   239
 CATAL?  Abs   25869 #0650D -   266   233
 CATCNT  Abs   26175 #0663F -   396   386
 CATCRD  Ext                -   173
 CATE34  Abs   26104 #065F8 -   368   374
=CATEDT  Abs   25653 #06435 -   182
 CATEXT  Abs   26059 #065CB -   349
 CATHDR  Abs   25517 #063AD -   159   225
 CATNK   Abs   25611 #0640B -   168   210
 CATP    Ext                -   176
 CATRPT  Abs   25816 #064D8 -   246   242
 CHKPSF  Ext                -  1167  1241  2289
 CK"ON"  Ext                -  1631
 CKINFO  Abs   99650 #18542 -    13  1685
 CLLINK  Ext                -  1179
 CLMPR1  Abs   28411 #06EFB -  2674  2682
=CLMPRO  Abs   28415 #06EFF -  2675
 CLPSTK  Ext                -  1178
 CRLFND  Ext                -   449
 CT$CRD  Ext                -   699
 CTALLP  Abs   26128 #06610 -   381   261   266
 CURDVC  Ext                -  2280
 CURRL   Abs  194536 #2F7E8 -    12  1254
 CURRST  Abs  193885 #2F55D -    12   892  1899
 D'LT04  Abs   27024 #06990 -  1182  1177
=D'LTE   Abs   26974 #0695E -  1160
 D'LTP   Ext                -  1158
 DO=GSB  Abs   27869 #06CDD -  2259  2302  2359  2379  2387  2432
 DOASCI  Ext                -  1022
 DOOUTB  Ext                -   784
 D1=AVE  Abs   99921 #18651 -    13   664   783
=D1=CRS  Abs   26755 #06883 -   892   182   705  1175  1572
 DISPt   Abs       0 #00000 -    11  1569
 DSPDLY  Ext                -   188
 DSPLIN  Ext                -  1283
 DVCNFE  Ext                -   694
 EDIT80  Ext                -   311
```

```
 ENDALL  Ext                  -  2318
=EOFLC+  Abs   28398 #06EEE  -  2629   218
=EOFLC1  Abs   28401 #06EF1  -  2630
=EOFLCH  Abs   28394 #06EEA  -  2628  2632
 EOLSN5  Ext                  -  2456
 EXPRj   Ext                  -   792
 F-RO-O  Abs  194715 #2F89B  -    12   796
 FILBLK  Abs   28310 #06E96  -  2507  1171  2277
=FILSK+  Abs   28445 #06F1D  -  2730
=FILSKP  Abs   28443 #06F1B  -  2729   661  2628
 FILXQ$  Ext                  -   624
 FINDA   Ext                  -   249
 FINDF+  Ext                  -   329  1585  2241
 FINDL+  Ext                  -  1268
 FINDLO  Ext                  -  1739
 FINDLR  Ext                  -  1255
 FLADDR  Ext                  -  2281
=FLSKPB  Abs   28441 #06F19  -  2728  2674
=FSPECj  Abs   27794 #06C92  -  2232   326  1582  2238
 FSPECx  Ext                  -  2232
 FTYP45  Abs   26927 #0692F  -  1092  1084
 FTYP55  Abs   26940 #0693C  -  1098  1077
=FTYPD+  Abs   26871 #068F7  -  1072
=FTYPDC  Abs   26882 #06902  -  1076   739
 FTYPFD  Ext                  -  1076
=GETPR   Abs   27640 #06BF8  -  1901
=GETPR+  Abs   27637 #06BF5  -  1900
=GETPR1  Abs   27643 #06BFB  -  1903   728  1597  2249  2680
 GETPR2  Abs   27646 #06BFE  -  1904
 GETPR3  Abs   27668 #06C14  -  1913  1912
=GETPRO  Abs   27630 #06BEE  -  1899
 GETRG+  Ext                  -   637
 GETST*  Ext                  -  2275
 GETST1  Ext                  -  1603
 GSBSTK  Abs  193955 #2F5A3  -    12  2259  2321
 HEXDEC  Ext                  -   849  1020
=INVFSP  Abs   26364 #066FC  -   675   686
 KMEMCK  Ext                  -  2358
 KYARGS  Ext                  -  2335
 KYFND+  Ext                  -  2342
 KYMRG+  Ext                  -  2386
 LDCM10  Ext                  -  1611
 LDCSET  Ext                  -   696  1277
 LINHA+  Ext                  -  1278
=LIST    Abs   27254 #06A76  -  1569
=LIST*   Abs   27258 #06A7A  -  1570  1546
 LIST10  Abs   27299 #06AA3  -  1582
 LIST20  Abs   27306 #06AAA  -  1585  1580
 LIST30  Abs   27219 #06A53  -  1553
 LIST50  Abs   27315 #06AB3  -  1592  1574  1577
 LIST60  Abs   27326 #06ABE  -  1595
 LIST65  Abs   27338 #06ACA  -  1600
 LIST75  Abs   27369 #06AE9  -  1611  1633
 LIST80  Abs   27382 #06AF6  -  1621  1609
 LISTDC  Ext                  -  1542  1566
```

```
 LISTP   Ext                  -  1543  1567
 LNARGS  Abs   27490 #06B62 -  1741  1608  2507
 LOCADR  Ext                  -   769
 LSTDON  Abs   27411 #06B13 -  1635  1625  1697  1749  1752
 LSTER   Abs   27223 #06A57 -  1555  1550  1583  1598
 LSTEXT  Abs   27208 #06A48 -  1549  1586
 LSTKEY  Ext                  -  1564
=LSTLEN  Abs   27687 #06C27 -  1964
=LSTLN+  Abs   27678 #06C1E -  1962  1281
 LSTLOK  Abs   27732 #06C54 -  1980  1971
 LSTPOL  Abs   27227 #06A5B -  1559  1593
 LSTXKY  Abs   27237 #06A65 -  1564  1596
 MERG04  Abs   27850 #06CCA -  2250  2287
 MERG06  Abs   27853 #06CCD -  2252
 MERG30  Abs   27911 #06D07 -  2275  2256
 MERG50  Abs   27960 #06D38 -  2293  2311
 MERG59  Abs   28024 #06D78 -  2316  2295
 MERG60  Abs   28035 #06D83 -  2320  2317
 MERG61  Abs   28041 #06D89 -  2321  2377
 MERG62  Abs   28072 #06DA8 -  2334  2257
 MERG65  Abs   28111 #06DCF -  2348  2344
 MERG67  Abs   28118 #06DD6 -  2354  2337
 MERG69  Abs   28172 #06E0C -  2381  2361
 MERGDC  Ext                  -  2236
=MERGE   Abs   27814 #06CA6 -  2238
 MERGEP  Ext                  -  2237
 MFERR   Ext                  -  1182
 MFWRQ8  Ext                  -  1974
 MLFFLG  Abs  194672 #2F870 -    12  1570
 MOVEU2  Ext                  -  2305
 MOVEU4  Ext                  -  2393
 MRGERR  Abs   27790 #06C8E -  2230  2213  2223  2239  2250  2683
 MRGEXT  Abs   27751 #06C67 -  2212  2242
 MRGFIN  Abs   27800 #06C98 -  2234  2226
 MRGKEY  Abs   28144 #06DF0 -  2365  2395
 MRGPOL  Abs   27766 #06C76 -  2219  2248
 MRGUP1  Abs   28236 #06E4C -  2440
 MRGUPD  Abs   28218 #06E3A -  2432  2293  2365
 MVMEM+  Ext                  -  1176
 NOSCRL  Ext                  -   313
 NXTLIN  Ext                  -  2296  2516
 NXTSTM  Ext                  -  1181
 OBCOLL  Ext                  -  1985
 OBLCMP  Ext                  -  1966
 OUTBS   Abs  193935 #2F58F -    12  1688  2298
 OUTBYT  Ext                  -  1962
 OUTNBS  Ext                  -   715  1073
 PARMXQ  Abs   27527 #06B87 -  1808  1160  1594  2252
=PCEXPR  Abs   26664 #06828 -   791   665
 PEDITM  Ext                  -  2310
=PLIST   Abs   27200 #06A40 -  1545
 POPBUF  Ext                  -   316
 POPSTK  Ext                  -  2329
 Poll;   Ext                  -  1756
 PRINTt  Abs       1 #00001 -    11  1545
```

```
 PRMSV    Abs   28238 #06E4E -  2442  2336  2508
 PRMSV2   Abs   28298 #06E8A -  2465  2448
 PRMSV5   Abs   28340 #06EB4 -  2519  2514
 PRMSV7   Abs   28344 #06EB8 -  2521  2350
 PRMXQ0   Abs   27549 #06B9D -  1816  1240
 PRMXQ1   Abs   27588 #06BC4 -  1832  1827
=PRPSND   Abs   27415 #06B17 -  1683  1613
 PRT#D3   Abs   26747 #0687B -   856   844
=PRT#DC   Abs   26689 #06841 -   837   775
 PSHADR   Abs   27890 #06CF2 -  2266  2278  2355
 PSHUPD   Ext                -  2272
 PURGDC   Ext                -   175
=RDENTY   Abs   27173 #06A25 -  1316  1098
 RMEMCH   Ext                -  2286
 ROMF-1   Ext                -   373   688
 RPTKY    Ext                -   241
=RSTD0    Abs   26674 #06832 -   796   791
 S-R0-0   Abs  194673 #2F871 -    12   246
 S-R0-1   Abs  194678 #2F876 -    12   215   228   287
 S-R0-2   Abs  194683 #2F87B -    12   220   318
 S-R0-3   Abs  194688 #2F880 -    12   154   381
 S-R1-1   Abs  194694 #2F886 -    12  1606  1694
 SAVD0    Ext                -   615
 SCRLLR   Ext                -   243
 SENDEL   Abs   97729 #17DC1 -    13  1692
 SNDWD+   Abs   97823 #17E1F -    13  1691
=WSR0-3   Abs   25504 #063A0 -   154   201   364
 ZERPGM   Ext                -  2320
 ave=d1   Ext                -   623   640
 bserr    Abs   26079 #065DF -   359   327   339   678  1555  2230
 cat17    Abs   25905 #06531 -   283   293
 catcrd   Abs   25627 #0641B -   173   355
 ct$crd   Abs   26422 #06736 -   699   683
 dcplin   Ext                -  1287
 eFPROT   Ext                -  1914
 eFSPEC   Ext                -   675
 eFTYPE   Ext                -  2228
 eL2LNG   Ext                -  1973
 eolxc+   Ext                -   177  1573
 fBASIC   Abs   57876 #0E214 -    11  2567
 fKEY     Abs   57868 #0E20C -    11  2572
 findl0   Abs   27479 #06B57 -  1737  1745  2513
 fltdh    Ext                -   643
 invarg   Abs   27624 #06BE8 -  1846  1843
 k#BOT    Ext                -   256
 k#DOWN   Ext                -   250
 k#TOP    Ext                -   254
 k#UP     Ext                -   252
 lDATEh   Abs       6 #00006 -    11   742   761
 lEOL     Abs       2 #00002 -    11  1741
 lFNAMh   Abs      16 #00010 -    11   724   725   297   298   726   740
 lFTYPh   Abs       4 #00004 -    11  1904  1906
 lTIMEh   Abs       4 #00004 -    11   761
 nxtstm   Abs   27018 #0698A -  1181   196   314  1635  2234  2330
 oFLENh   Abs      32 #00020 -    11   271   298   740  1600  2731
```

oFTYPh	Abs	16	#00010	-	11	1600	1903	2562	2566		
obcoll	Abs	27745	#06C61	-	1985	447	788				
pCAT	Abs	6	#00006	-	11	350					
pCAT$	Abs	7	#00007	-	11	670					
pLIST	Abs	12	#0000C	-	11	1552					
pLIST2	Abs	46	#0002E	-	11	1560					
pMERGE	Abs	13	#0000D	-	11	2215					
pMRGE2	Abs	47	#0002F	-	11	2222					
pPOLL2	Ext			-	351						
=pPOLL3	Abs	27776	#06C80	-	2223	1562					
poll	Abs	27521	#06B81	-	1756	349	669	1551	1559	2214	2221
popbuf	Abs	25993	#06589	-	316	185	240	308	393		
pshupd	Abs	27905	#06D01	-	2272	2269					
romchk	Ext			-	365	397					
romfnd	Ext			-	387						
savel0	Ext			-	1276						
tALL	Ext			-	203						
tCARD	Ext			-	168						
tCOMMA	Ext			-	1575	1817					
tEOL	Ext			-	2455						
tKYSck	Ext			-	209	1579					

Input Parameters

  Source file name is SG&SYS::MS

  Listing file name is SG/SYS:TI:ML::-1

  Object file name is SG%SYS:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                      STITLE RUN/CONT/CHAIN
 2                *
 3                *        J  PPPP    &      SSS   Y   Y   SSS
 4                *        J  P   P  & &    S   S  Y   Y  S   S
 5                *        J  P   P  & &    S       Y Y   S
 6                *        J  PPPP    &      SSS     Y     SSS
 7                *        J  P      & & &       S    Y        S
 8                *     J  J  P      &  &   S   S    Y    S   S
 9                *      JJJ   P      && &   SSS     Y     SSS
10                *
11                      TITLE  System Commands, RUN Loop<831212.1204>
12 06F2F         JPSYS  ABS    #06F2F
13                      RDSYMB TI%EQU::MS
14                      RDSYMB SB%RAM::MS
15                ********************************************************************
16                ********************************************************************
17                **
18                ** Name:    RUN     -  Statement Execution
19                ** Name:    CONT    -  Statement Execution
20                ** Name:    CHAIN   -  Statement Execution
21                ** Name:    EXITRN  -  Statement Execution
22                **
23                ** Category:   STEXEC
24                **
25                ** Purpose:
26                **      Execute RUN and CONT Statements and Keys
27                **      Execute CHAIN statement
28                **      Exit from program (BASIC/Binary)
29                **
30                ** Entry:
31                **      RUN:    DO past RUN token
32                **      CONT:   DO past CONT token
33                **      CONTK:  CONT key entered
34                **      RUNK:   RUN key entered
35                **      CHAIN:  DO past CHAIN token
36                **      EXITRN: Clears PRGM annunc,prgm flags, exits BASIC
37                **
38                ** Exit:
39                **      Through BSCEX2
40                **        Run from DO
41                **      Through NXTSTM
42                **        If Null Program   (NOP)
43                **           or Line# not found
44                **      Through CRGJMP,EXITRN
45                **        If Binary program
46                **
47                **      Error Exits - Through BSERR 'cus of POLLing
48                **        eSTMNF      Statement not found
49                **                    Line# not found (if not in PRGMSCOP
50                **        eFTYPE      Illegal File Type
51                **                     from PRSCOP of Current File
52                **                     If RUN File # BASIC
53                **        eFSPEC      Illegal File Specification
54                **        eFnFND      File to RUN not found
55                **        eMEM        Not enough memory to save file info
```

```
56          **                    Not enough memory to copy in file
57          **        eFEXST      File to COPY/RUN exist in main memory
58          **
59          ** Calls:      PRSCOO,AUTCLR,FINDLR,LBLRCK,FINDLB,NULLP,
60          **             CHAIN+,GETSTC,CRLFOF,SCOPCK,EDIT20,FSPECx,COPYu,
61          **             FINDF, SFGPGM,DO=PCA,COLLAP,PRGFMF,CURDVC,SAVPCr,
62          **             RDHDR1,BASCHK,POLL,DLFIBA,STMBCL,CNTCHK,ALINFO,
63          **             SVINFO,CLPSTK
64          **
65          ** Uses:       A-D,CNTADR,D0,D1,S13,sCONTK (S9),sCONT (S10)
66          **             sCHAIN (S11),S-R0-1,S-R0-2,sDEST (S3),sREADI(S4),
67          **             sRUNBn (S4), sMAINc (S5)
68          **             R0-R3,All of FUNCTION scratch,STMTD0,STMTR1,
69          **             S1,S2,S6,S8,S9,S7,SCRTCH (32 nibs),S-R1-0
70          **
71          **             S-R0-1    = Saved D0
72          **             STMTD0    = Save status (COPYu uses STMTR0&1)
73          **
74          ** Stk lvls:   7
75          **
76          ** Detail:    CONT [ <lineno> | <label> ]
77          **            RUN [ <file spec> ] [, <lineno> | <label> ]
78          **            RUN <lineno>
79          **            CHAIN <file spec>
80          **
81          ** Algorithm:
82          **
83          **  CHAIN:    Set CHAIN flag                          (sCHAIN)
84          **            goto 0.2;
85          **  CONTK:    Set CONT flag                           (sCONT)
86          **            goto 0;
87          **   RUNK:    Clear CONT flag
88          **     0:     Set CONT | RUN Key                      (sCONTK)
89          **            Send Carriage Return/ Line Feed         (CRLFOF)
90          **            goto 1.5;
91          **   CONT:    Set CONT flag                           (sCONT)
92          **            goto 1;
93          **    RUN:    Clear CHAIN flag                        (sCHAIN)
94          **    0.2:    Clear CONT and CONT Key flags           (sCONT,sCONTK
95          **            If File Spec follows                    (tRFILE)
96          **               Skip token
97          **               Execute File Specification           (FSPECx)
98          **               If Illegal File spec                 (Carry set)
99          **                  Error --> Illegal File Spec       (eFSPEC)
100         **               else
101         **                  Save filename                     (R1)
102         **                  Save Prgm Counter - relative      (SAVPCr)
103         **                  Clear Dest flag and MAIN Chain flag
104         **                  If no device spec & CHAIN
105         **                     Set MAIN Chain flag
106         **                  If External Device | CARD | Ext CHAIN stmt
107         **     0.3:            Allocate Save area             (ALINFO)
108         **                     Save Source information        (SVINFO)
109         **                     Set Dest Device = MAIN
110         **                     Save Destination inform        (SVINFO)
```

```
111             **                  External Device flag = sDEST
112             **               If CHAIN
113             **                  Save status                  (RSTK)
114             **                  Get current file device      (CURDVC)
115             **                  Purge current file           (PRGFMF)
116             **                     Error if carry set
117             **                  Restore Status
118             **               If External Device              (sMAINc)
119             **                  Copy file into mainframe     (COPYu)
120             **                  Set D1 @ File start
121             **               else
122             **                  If MAIN Chain
123             **                     Restore filename from SAVSTK
124             **                  else
125             **                     Restore filename from R1
126             **                  Find file                    (FINDF)
127             **                  If not found --->            (eFnFND)
128             **                     If CHAIN
129             **                        Release SAVE Stack area (CPYEXT)
130             **                     Error Exit
131             **     0.7:      Clear RUN Binary flag           (sRUNBn)
132             **               Read file header                (RDHDR1)
133             **     0.8:      If not BASIC file               (BASCHK)
134             **                  If not Binary
135             **                     POLL for Run w/ Unknown File Type (POLL)
136             **                     Error Exit if Return     (RUNERR)
137             **                  Set RUN Binary flag         (sRUNBn)
138             **               Reset CURRST/EN                 (EDIT20)
139             **               goto 2;
140             **      1:    Save Prgm Counter - relative       (SAVPCr)
141             **    1.5:    Clear RUN Binary flag              (sRUNBn)
142             **            Read Current file header           (RDCHDR)
143             **            If not BASIC file                  (BASCHK)
144             **               goto 0.8
145             **      2:    If not (CONT and CNTADR#0)
146             **               Pos to file start w/o Ftype Chk (GETSTC)
147             **               Collapse stacks/zero prgm menro (CLPSTK)
148             **               Set program scope;              (PRSC00)
149             **               Delete FIB, collapse ASSIGN table (DLFIBA)
150             **               Chain current program           (CHAIN+)
151             **            Clear AUTO mode                    (AUTCLR)
152             **            If RUN Binary
153             **               Position to first line of file  (GETSTC)
154             **               goto 7;
155             **            If CONT | RUN Key                  (sCONTK)
156             **               Run with no parameters          (goto 5)
157             **            If CHAIN
158             **               Run with no parameters          (goto 5)
159             **            Restore D0                         (S-R0-2)
160             **            Read and skip token
161             **            If comma
162             **               Read and skip next token
163             **            If line#
164             **               Skip Jump address and Read line#
165             **               Find line# across Program file  (FINDLR)
```

```
166             **              If found                                (Carry set)
167             **                 If line# NOT within Program          (SCOPCK)
168             **                    Error --> Line# not found
169             **                 Position to EOL | @   (DO <-- D1-2)
170             **                 goto 6;
171             **              else
172             **                 If line# > found                     (S1=0)
173             **                    goto 6;
174             **                 else
175             **      3:            Do nothing/Return                 (NXTSTM)
176             **           Check if Label                             (LBLRCK)
177             **              If bad label    (Carry set)
178             **      4:         Error --> Label not found
179             **              If label reference   (S8=1)
180             **                 Find label within Program Scope(FINDLB)
181             **                 If found
182             **                    Collapse Math stack              (COLLAP)
183             **                    goto 6;
184             **                 else
185             **                    goto 4;
186             **      5:    No Parameters
187             **               If NULLP ---> Return                   (goto 3)
188             **               If CONT                                (sCONT)
189             **                  If CNTADR # 0
190             **                     DO <-- CNTADR
191             **                     goto 6;
192             **               Get first line of program              (GETSTC)
193             **      6:    If CONT
194             **              Preserve DO
195             **              Collapse Statement Buffer              (STMBCL)
196             **              Restore DO
197             **      7:    Clear SUSP Annunc                        (CLSUSP)
198             **            Set PRGM Annunc, PgmRun flag             (SFGPGM)
199             **            If RUN Binary
200             **               Load Binary File type into A          (fBIN)
201             **               Poll for RUN non BASIC file           (POLL)
202             **               Jump  to start binary address
203             **            else
204             **               Set Program Running flag
205             **               go RUN Basic                          (RUNEX2)
206             **
207             **
208             ** History:
209             **
210             **   Date     Programmer          Modification
211             ** --------   ----------   ------------------------------------
212             ** 10/20/82   JP           Commented CHAIN/RUN external code
213             ** 11/19/82   JP           Added back/rewrote CHAIN/RUN code
214             ** 11/22/82   JP           Error Exit to BSERR, Error#=LC(4)
215             ** 12/18/82   JP           CHAIN+ call change, CLSUSP call
216             ** 01/19/83   JP           Moved CLPSTK/DELFIB @ RUN075
217             ** 03/18/83   JP           Packed CLSUSP 'cus SFGPGM does it now
218             ** 03/23/83   JP           Using S-R0-1 instead of STMTR0+64,
219             ** 03/23/83   JP           Using S-R0-2 instead of PRMCNT
220             ** 03/23/83   JP           Changed sCONTK,sCONT to S9,S10
```

```
221             ** 03/24/83   JP          CHAIN has no [<statement id> ]
222             ** 04/04/83   JP          eFSPEC if no filename with RUN
223             ** 04/05/83   JP          Allow no filename if :CARD/PCRD
224             ** 04/21/83   JP          CLRST after Binary return
225             ** 04/21/83   JP          Don't clear PRGM,PgmRun after Binary
226             ** 04/26/83   JP          If PgmRun, check exceptions@RUNRT1
227             ** 05/02/83   JP          Packed explicit status clear @ RUN
228             ** 05/02/83   JP          CHAIN w/ error must CLPSTK,clr PgmRun
229             ** 05/05/83   JP          CHAIN w/ ALL errors;CLPSTK,clr PgmRun
230             ** 06/05/83   JP          RTN to Binary program; don't CRGJMP
231             ** 06/10/83   JP          CHAIN doesn't close open files
232             ** 06/30/83   JP          Collapse Math Stack after LABEL eval
233             ** 08/11/83   JP          Use STMTDO instead of S-RO-2 for Statu
234             **                        SR# 998-5,999-3
235             ** 08/12/83   JP          Clear PgmRun after Purge Current file
236             **                        for CHAIN: SR#997-7
237             **
238             ************************************************************************
239             ************************************************************************
240             ************************************************************************
241             ************************************************************************
242             **
243             ** Name:(S) pRUNft  -  Poll on RUN with unknown filetype
244             **
245             ** Category:   POLL
246             **
247             ** Type:       POLL
248             **
249             ** Purpose:
250             **      Poll on RUN with file of "unknown" filetype
251             **      Filetype is NOT BASIC or Binary
252             **
253             ** Should poll be "Handled" (return with XM=0)?:
254             **      No - this is a take-over Poll
255             **
256             ** Meaning of "Handling" Poll (what does code do if handled?):
257             **      Take over the RUN execution of the file
258             **
259             ** Entry conditions for handler (registers, ST, RAM, etc.):
260             **      B[A] = Poll number (pRUNft)
261             **      HEX mode.
262             **      P=0.
263             **      A(A) = File type  (HEX)
264             **      R2(A)= File type  (HEX)
265             **      D1   @ File length (offset) field in file header
266             **             5 nibble read @ D1 = Offset to next file
267             **             Current D1+ offset --> Next file start
268             **
269             **             CAT file length =
270             **               File offset value - Offset to data of file
271             **
272             **      sCONT  = 1 if CONT            (S10)
273             **      sCONTK = 1 if CONT/RUN key    (S9)
274             **      sCHAIN = 1 if CHAIN statement (S11)
275             **
```

```
276             ** Normal exit conditions from handler if handled (ST, RAM,
277             ** registers, etc.):
278             **      Carry clear
279             **      HEX mode.
280             **      XM=0.
281             **      GOVLNG to MAIN05
282             **      or
283             **      GOVLNG to BSCEXT to exit the BASIC interpreter
284             **        This is done by BASIC and Binary programs
285             **        Filetype read, Buffers are flushed
286             **        A fast poll is issued: pBSCex
287             **
288             **      See NOTE below
289             **
290             **
291             ** Normal exit conditions from handler if not handled (ST, RAM,
292             ** registers, etc.):
293             **      Carry clear
294             **      HEX mode.
295             **      XM=1.
296             **      Preserve Status
297             **
298             ** Error exit conditions from handler
299             **      Error returns are ignored.
300             **      If the POLL returns:
301             **         If carry set ---> "eMEM" from POLL
302             **            else      ---> "eFTYPE" from RUN
303             **
304             ** Available subroutine levels: 7
305             **      --POLL handler is one level shallower than caller--
306             **      RUN is a top level statement/command
307             **
308             ** NOTE:
309             **      Any Lex File running a non BASIC file should:
310             **          Clear the SUSP annunciator
311             **          Set the PRGM annunciator    (see SFGPGM)
312             **          Collapse all BASIC stacks   (see CLPSTK)
313             **          Set CURRST, CURREN @ file   (see EDIT20)
314             **
315             **      Responder should issue a pRUNnB (RUN non BASIC) Poll
316             **          The mainframe issues a "pRUNnB" when
317             **          running a Binary file with the filetype in A(A)
318             **
319             ** What registers/RAM may be used if handled?:
320             **      --A-D, D0, D1, P always available--
321             **      RUN is in complete control at this point
322             **
323             ** What registers/RAM may be used if not handled?:
324             **      --A-D, D0, D1, P always available
325             **      Global status (S12-S15) are sacred
326             **
327             ** What registers/RAM may be used if error exit?:
328             **      No error exit allowed
329             **
330             ** Envisioned application(s):
```

```
331             **      Extend RUN statement to handle other file types
332             **
333             ** History:
334             **
335             **    Date     Programmer            Modification
336             **  --------   ----------   ----------------------------------
337             **  09/16/82   JP           Added Poll
338             **  01/16/83   JP           Check carry from Poll
339             **  04/19/83   JP           Updated Documentation
340             **  04/24/83   JP           Changed entry conditions
341             **
342             ********************************************************************
343             ********************************************************************
344             ********************************************************************
345             ********************************************************************
346             **
347             ** Name:(S) pRUNnB  -  Poll before non BASIC file exec (BIN)
348             **
349             ** Category:   POLL
350             **
351             ** Type:       POLL
352             **
353             ** Purpose:
354             **      Poll before starting execution or continuing execution
355             **      of a non BASIC file
356             **
357             **      Poll before running a BINARY file
358             **
359             **
360             ** Should poll be "Handled" (return with XM=0)?:
361             **      No - let this poll go to all other Lex files
362             **
363             ** Meaning of "Handling" Poll (what does code do if handled?):
364             **      Perform any "system" or special initialization needed
365             **      before Binary file is executed.
366             **
367             **
368             ** Entry conditions for handler (registers, ST, RAM, etc.):
369             **      B[A] = Poll number (pRUNnB)
370             **      HEX mode.
371             **      P=0.
372             **
373             **      R2(A)= File type of file to execute
374             **      DO   @ Start of code to execute
375             **      BASIC stacks have been collapsed
376             **
377             **      General purpose poll
378             **
379             **      Mainframe poll will ALWAYS be Binary exeucute
380             **        R2(A)= fBIN
381             **        DO   @ Start of binary file
382             **        If sCONT (S10) = 1
383             **           Executing a CONT statement
384             **           CNTADR is always zero, unless a Lex file has
385             **             set this  (see pBSCex Poll)
```

```
386             **          Therefore, CONT is always a RUN
387             **        If sCONTK (S9) = 1
388             **          RUN or CONTK hit
389             **        If sCHAIN (S11)= 1
390             **          CHAIN statement
391             **        SUSP annunciator has been cleared
392             **        PRGM annuciator and PgmRun flag have been set
393             **        Current file pointers @ Binary file
394             **
395             ** Normal exit conditions from handler if handled
396             **
397             **      This poll should NOT be indicated as handled so
398             **      other Lex files can "set-up" before execution.
399             **
400             **      If Lex file wants to be the ONLY Lex file to handle:
401             **      then:
402             **
403             **      Carry clear
404             **      HEX mode.
405             **      XM=0.
406             **      D0 must be PRESERVED!!!
407             **      Preserve S3
408             **
409             ** Normal exit conditions from handler if not handled
410             **
411             **      Carry clear
412             **      HEX mode.
413             **      XM=1.
414             **      Status intact: sCONT(10), sCONTK(S9), sCHAIN(S11),S3
415             **
416             ** Error exit conditions from handler:
417             **      Error return has no meaning ---
418             **
419             ** Available subroutine levels:
420             **      --POLL handler is one level shallower than caller--
421             **
422             **      This is a RUN... therefore all levels (6) available
423             **      Must be able to return to POLL routine
424             **
425             ** NOTE:
426             **   See Special memory/pointer considerations
427             ** What registers/RAM may be used if handled?:
428             **      --A-D, D0, D1, P always available--
429             **      --R0-R4, scratch RAM?--
430             **
431             ** What registers/RAM may be used if not handled?:
432             **      --A-D, D0, D1, P always available
433             **      --R0-R4, scratch RAM?--
434             **
435             ** What registers/RAM may be used if error exit (POLL only)?:
436             **      No error exit allowed
437             **
438             ** Special memory/pointer considerations (are pointers funny?):
439             **
440             ** Binary Files will always be RUN/CONT from the start of
```

```
441             **   the file... it is "impossible" to systematically
442             **   return a meaningful CONTinue address through the BASIC
443             **   loop.  If a Binary file wishes to implement CONT...
444             **   it should respond to the pBSCex poll:
445             **      If current filetype is Binary and sERROR=1
446             **         Update CNTADR @ Binary code to CONTinue at
447             **         Set the SUSP annunciator (SFLAGs)
448             **
449             ** If a Poll Handler intends to CALL BASIC from within:
450             **   Return Address to Poll must be saved on the GOSUB stack
451             **     (Use PSHUPD and POPUPD)
452             **   The FORSTK must be adjust OVER the Poll Save information
453             **     before the CALL and readjust after.  CALL uses the
454             **     FORSTK pointer to save information and if FORSTK is
455             **     not adjusted, Poll Information is overwritten
456             **
457             **     Before CALL                After CALL
458             **
459             **     D1=(5)  =FORSTK              "
460             **     A=DAT1  A                    "
461             **     C=0     A                    "
462             **     LC(2)   =1POLSV              "
463             **     A=A-C   A                   A=A+C   A
464             **     DAT1=A  A                   DAT1=A  A
465             **
466             ** Envisioned application(s):
467             **      Set up I/O buffers before Binary execution or some
468             **      type of non-BASIC file.
469             **
470             ** History:
471             **
472             **    Date     Programmer              Modification
473             **  --------   ----------   ------------------------------------
474             **  09/16/82   JP           Added Poll
475             **  01/16/83   JP           Generalized Poll for any file type
476             **  04/22/83   JP           Upgraded documentation
477             **  04/25/83   JP           Pass File type in R2 instead of A
478             **
479             ********************************************************************
480             ********************************************************************
481             *
482             * CHAIN Entry
483             *
484 06F2F 0000         REL(5) =RUNDC         CHAIN Decompile
        0
485 06F34 0000         REL(5) =CHAINP        CHAIN Parse
        0
486 06F39 858  =CHAIN  ST=1   sCHAIN         Set CHAIN flag
487 06F3C 6130         GOTO   RUN015
488             *
489             * CONT Key Entry
490             *
491 06F40 85A  =CONTK  ST=1   sCONT          Set CONT flag
492 06F43 6600         GOTO   RUN005
493             *
```

```
                        * RUN Key Entry
                        *
06F47 84A  =RUNK   ST=0   sCONT          Clear CONT flag
06F4A 859   RUN005 ST=1   sCONTK         Set CONT/RUN Key flag
06F4D 7CC3         GOSUB  CrLfof         Send CR/LF to clear display
06F51 6371         GOTO   RUN070
                        *
                        * CONT Entry
                        *   Must clear Statement Buffer if CONT entered
                        *   This prevents a return to the statement buffer, when continui
                        *        a program initiated from the keyboard, with a prior stat
                        *   A null statement buffer will prevent the return to keyboard
                        *
06F55 0000         REL(5) =CONTP         CONT Parse address
      0
06F5A 85A  =CONT   ST=1   sCONT          Set CONT flag
06F5D 849          ST=0   sCONTK         Clear RUN/CONT Key flag
06F60 6061  RUN010 GOTO   RUN065
                        *
                        * RUN Entry
                        *   Status are clear on entry
                        *
06F64 0000         REL(5) =RUNDC         RUN Decompile
      0
06F69 0000         REL(5) =RUNP          RUN Parse
      0
06F6E      =RUN
06F6E 14A   RUN015 A=DAT0 B
06F71 3100         LC(2)  =tRFILE        Run File Spec
06F75 966          ?A#C   B              No RUN File Spec ?
06F78 8E           GOYES  RUN010
                        *
                        * RUN <filespec>
                        *
06F7A 161          D0=D0+ 2              Skip tRFILE
06F7D 7000         GOSUB  =FSPECj        Execute File Specification
06F81 541          GONC   RUN020         Good File Spec
06F84 8C00  RUNERR GOLONG =BSERR         Error Exit
      00
                        *
                        * JUMP Back from RUN020 code:
                        *
                        * If CHAIN and NO device specified
                        *    Set MAIN Chain flag and purge current file
                        * else
                        *    Restore filename
                        *    go RUN in MAIN
                        *
06F8A 87B   RUN017 ?ST=1  sCHAIN         CHAIN ?
06F8D 55           GOYES  RUN034         Set MAIN Chain flag
06F8F 111   RUN018 A=R1                  Restore filename
06F92 6DC0         GOTO   RUN053         RUN in MAIN
                        *
                        * If not CARD | PCRD device
                        *    If no filename
```

```
                *       Error exit ---> eFSPEC
                * Save filename (R1)
                * Save D0 - relative position
                *
06F96 2F    RUNO20 P=     15
06F98 307          LC(1)  =dCARD
06F9B 20           P=     0
06F9D 943          ?C=D   S              CARD device ?
06FA0 D0           GOYES  RUNO22         Don't check filename
06FA2 97C          ?A#0   W              Filename ?
06FA5 80           GOYES  RUNO22
06FA7 8C00         GOLONG =FSPCER        eFSPEC if no filename
      00
06FAD 101   RUNO22 R1=A                  Save filename
06FB0 7D52         GOSUB  SAVPCr         Save PC - Relative Position
06FB4 843          ST=0   sDEST          Clear destination flag
06FB7 845          ST=0   sMAINc         Clear MAIN Chain flag
06FBA ACB          C=D    S              D(S) = Device
06FBD B46          C=C+1  S              Check for NO Device C(S)=F
06FC0 49C          GOC    RUNO17         Mainframe Run/CHAIN
                *
                * If external device | PORT & Chain
                *    Allocate save area
                *    Restore filename
                *    Save Source file information
                *    Save Destination file information
                *    sDEST === External Device flag
                *
06FC3 A4E   RUNO25 C=C-1  S              Restore Device Type
06FC6 A46          C=C+C  S              Check for External Device
06FC9 4B1          GOC    RUNO35         External Device
06FCC 2F           P=     15             Check Device Type
06FCE 301          LC(1)  dPORT
06FD1 20           P=     0
06FD3 9C3          ?D>C   S              not MAIN or PORT ?
06FD6 F0           GOYES  RUNO35         External Device
06FD8 86B          ?ST=0  sCHAIN         not Chain ?
06FDB 4B           GOYES  RUNO18         RUN from "internal" device
06FDD 943          ?C=D   S              PORT ?
06FE0 50           GOYES  RUNO35         Not MAIN chain
                *
                * External Device
                *   sMAINc set if CHAIN with Mainframe Device type
                *
06FE2 855   RUNO34 ST=1   sMAINc         Set Mainframe CHAIN flag
06FE5 8E00  RUNO35 GOSUBL =ALINFO        Allocate save area
      00
06FEB 489   RUNER5 GOC    RUNERR         Error Return
06FEE 111          A=R1                  Restore filename
06FF1 8E00         GOSUBL =SVINF+        Save source information
      00
06FF7 AF0          A=0    W              Clear destination filename
06FFA D3           D=0    A              Destination device = MAIN
06FFC 853          ST=1   sDEST          Set destination flag
06FFF 8E00         GOSUBL =SVINFO        Save destination information
```

```
          00
    597           *
    598           * If CHAIN
    599           *    Save status
    600           *    Get current file device
    601           *    Purge current file
    602           *    Clear program running flag
    603           *      If error within COPYu
    604           *         Prevent SUSP into newly EDIT workfile
    605           *
    606 07005 86B  RUN040 ?ST=0  sCHAIN          not Chain ?
    607 07008 C1          GOYES  RUN045
    608 0700A 0B          CSTEX                  Save Status, cus PRGFMF destroys
    609 0700C 06          RSTK=C
    610 0700E 8E00        GOSUBL =CURDVC         Get current file device
          00
    611 07014 8E00        GOSUBL =PRGFMF         Purge current file
          00
    612 0701A 40D         GOC    RUNER5          Error return
    613 0701D 84D         ST=0   PgmRun          Clear program running flag
    614 07020 07          C=RSTK
    615 07022 0B          CSTEX                  Restore Status
    616           *
    617           * If external device (not Mainframe CHAIN)
    618           *    Save status
    619           *    Copy file into mainframe
    620           *    Restore status
    621           *    Set D1 @ file start
    622           *
    623 07024 875  RUN045 ?ST=1  sMAINc          MAINframe chain ?
    624 07027 03          GOYES  RUN050
    625 07029 0B   RUN047 CSTEX
    626 0702B 1B19        D0=(5) =STMTD0
          8F2
    627 07032 15C2        DAT0=C 3               Save Status 0-3
    628 07036 8E00        GOSUBL =COPYu          Copy file into mainframe
          00
    629 0703C 1B19        D0=(5) =STMTD0
          8F2
    630 07043 0A          ST=C                   Preserve C(0-2) = Error Num
    631 07045 15E2        C=DAT0 3
    632 07049 0B          CSTEX                  Restore status and C(0-2)
    633 0704B 405         GOC    RUNER8          Error return from COPY
    634 0704E 119         C=R1                   C @ file start
    635 07051 135         D1=C                   Set D1 @ file start
    636 07054 5E1         GONC   RUN055          B.E.T.
    637           *
    638           * Filename specified has no device or device=MAIN
    639           *
    640           *   If CHAIN
    641           *      Restore filename from SAVE area
    642           *      Shift device to D(S)
    643           *   Find File to RUN/CHAIN
    644           *   If CHAIN and PORT device
    645           *      Go copy file in and Run
```

```
646                *
647 07057 8E00     RUNO50 GOSUBL =RDINFS        Restore filename/device from SAVE
          00
648 0705D 817             DSRC                 Move device to D(S)
649 07060 8E00     RUNO53 GOSUBL =FINDF         Find file to RUN/CHAIN
          00
650 07066 453             GOC    RUNER8        File not found
651 07069 86B             ?ST=0  sCHAIN        not CHAIN ?
652 0706C 70              GOYES  RUNO55
653 0706E 94F             ?D#0   S             not MAIN ? (PORT)
654 07071 8B              GOYES  RUNO47
655                *
656                * D1 @ Start of file to execute
657                * Read file header w/ File Type  (P=0 from FINDF/COPYu)
658                * If non-BASIC file
659                *    If Binary file
660                *       Set RUN Binary file flag
661                *    else
662                *       POLL on RUN with unknown file type
663                *       Error Exit ---> eFTYPE if no response
664                *                       eMEM   if carry set
665                *
666 07073 844      RUNO55 ST=0   sRUNBn         Clear RUN Binary flag
667 07076 7386            GOSUB  RDHDR1
668 0707A 70C6            GOSUB  BASCHK        Check if BASIC file
669 0707E 533             GONC   RUNO60        A BASIC file
670 07081 3440     RUNO57 LC(5)  =fBIN
          2E0
671 07088 8A2             ?A=C   A             Binary file ?
672 0708B 42              GOYES  RUNO58
673 0708D 7756            GOSUB  Poll
674 07091 03              CON(2) =pRUNft       RUN with unknown File Type
675 07093 480             GOC    RUNER8        Memory Error ?
676 07096 3300            LC(4)  =eFTYPE       Illegal File Type
          00
677                *
678                * If CHAIN and error after PURGED current file
679                *       Perform ENDALL - Incase Prog was SUSP
680                *
681 0709C 86B      RUNER8 ?ST=0  sCHAIN         Not CHAIN ?
682 0709F C0              GOYES  RUNER9
683 070A1 06              RSTK=C               Save Error#
684 070A3 8E08            GOSUBL =CLPSTK       Collapse stacks,etall
          C0
685 070A9 07              C=RSTK               Restore Error#
686 070AB 68DE     RUNER9 GOTO   RUNERR        Must exit BSERR (due to COPYu)
687                *
688                * File specified is either BASIC or Binary
689                * Position to file start, Update CURRST and CURREN
690                *
691 070AF 854      RUNO58 ST=1   sRUNBn         Set RUN Binary flag
692 070B2 1CF      RUNO60 D1=D1- (oFLENh)-(oFTYPh)
693 070B5 1CF             D1=D1- (oFTYPh)-(oFNAMh) Position to File start
694 070B8 8E00            GOSUBL =EDIT20       Update CURRST/EN
          00
```

```
695 070BE 541          GONC   RUN075         B.E.T.
696             *
697             * Save DO
698             *
699 070C1 7C41  RUN065 GOSUB  SAVPCr          Save Program counter - Relative
700             *
701             * At CURRENT file
702             *    Check Filetype
703             *    If not BASIC --> go check for Binary or POLL
704             *
705             * NOTE:  If current file = BINARY
706             *           CURRST,CURREN are updated anyways to save code
707             *           Stacks are collapsed, also
708             *
709             * Clear AUTO Mode
710             *
711 070C5 844   RUN070 ST=0   sRUNBn          Clear RUN Binary flag
712 070C8 7226        GOSUB  RDCHD+          Read current file hdr w/ File type
713 070CC 7E66        GOSUB  BASCHK          Check if BASIC program
714 070D0 40B         GOC    RUN057          Not BASIC
715             *
716             * If not (CONT and non-zero CONTINUE address)
717             *    Get start of current file
718             *        GETSTC must be called, then PRSCOO
719             *        Since PRSCOP errors out if Filetype # BASIC
720             *        RUN Binary file falls through this code
721             *    Collapse stacks, zero program associated memory
722             *    Set Program Scope, PRGMST, PRGMEN
723             *    If not CHAIN
724             *       Delete FIB w/o flushing I/O buffers, collapse ASSIGN tabl
725             *    Chain current program
726             *
727             *****
728             * If CONT and non-zero Continue address, program scope is valid
729             *    and cannot be changed---could be suspended within SUB progra
730             *
731 070D3 7221  RUN075 GOSUB  CNTCHK          CONT and non-zero Continue addr ?
732 070D7 502         GONC   RUN775          yes - don't reset Program Scope
733 070DA 8E94        GOSUBL CLPSTK          Collapse stacks/zero prgm memory
          C0
734 070E0 8EA9        GOSUBL PRSCO-          Get program Scope w/o File type err
          A0
735 070E6 87B         ?ST=1  sCHAIN          CHAIN ?
736 070E9 90          GOYES  RUN773          Don't close files
737 070EB 8F00        GOSBVL =DLFIBA         Delete FIB, collapse ASSIGN table
          000
738 070F2 8EA1  RUN773 GOSUBL CHAIN+          Chain current program
          B0
739 070F8 8E00  RUN775 GOSUBL =AUTCLR         Clear AUTO Mode
          00
740 070FE 864         ?ST=0  sRUNBn          Not RUN Binary ?
741 07101 A0          GOYES  RUN076
742             *
743             * If RUN Binary
744             *    Position to first line of Binary file
```

```
745             *       (Filetype in R2 for pRUNnB Poll
746             *    Go set PRGM annunciator and RUN
747             *
748 07103 7F16          GOSUB  GETSTC          Position to execution address
749 07107 60D0          GOTO   RUN160
750             *
751             * If CONT/RUN key
752             *    go RUN with No parameters to check
753             * If CHAIN
754             *    go RUN with No parameters to check
755             *
756 0710B 879   RUN076 ?ST=1  sCONTK          CONT | RUN Key ?
757 0710E 70            GOYES  RUN077
758 07110 86B           ?ST=0  sCHAIN
759 07113 60            GOYES  RUN080          not CHAIN ?
760 07115 6B80  RUN077 GOTO   RUN120          CONT or RUN w/o parms
761             *
762             * Restore D0 - Program Counter
763             * Read and skip next token
764             * If comma  (from RUN)
765             *    Read & skip next token
766             * Look for Parameters
767             *
768 07119 1B67  RUN080 D0=(5) =S-R0-1
            8F2
769 07120 146           C=DAT0 A               Relative position of PC
770 07123 8E00          GOSUBL =D0=PCR
            00
771 07129 C2            C=C+A  A               Current D0
772 0712B 134           D0=C                   Restore D0
773 0712E 14A           A=DAT0 B               Read token
774 07131 161           D0=D0+ 2               Skip token
775 07134 3100          LC(2)  =tCOMMA
776 07138 966           ?A#C   B               not Comma ?
777 0713B 80            GOYES  RUN085
778 0713D 142           A=DAT0 A               Read next token
779 07140 161           D0=D0+ 2               Skip token
780 07143 3100  RUN085 LC(2)  =tLINE#
781 07147 966           ?A#C   B               not Line# ?
782 0714A 83            GOYES  RUN110
783             *
784             * <lineno>
785             *   Skip Jump Address
786             *   Read and Find line#
787             *
788 0714C 164           D0=D0+ 5               Skip jump address
789 0714F 8F00          GOSBVL =FINDLR         Read & Find line# across file
            000
790 07156 401           GOC    RUN090          Line# found
791             *
792             * If Line# # found
793             *    Return / Do nothing
794             * If Line# > found
795             *    go Check if line# within Program scope
796             *
```

```
797 07159 871          ?ST=1  1              LineN not found ?
798 0715C 70           GOYES  RUNEXT
799 0715E 860          ?ST=0  0              LineN > found ?
800 07161 60           GOYES  RUN090
801 07163 6175 RUNEXT  GOTO   Nxtstm         Return/ Do nothing
802               *
803               * If line found NOT within Program Scope
804               *    Error Exit
805               *
806 07167 133  RUN090  AD1EX                  Move LineN address to A
807 0716A 7455         GOSUB  Scopck         Check if within Program Scope
808 0716E 4C0          GOC    RUNER1         Not Within scope
809               *
810               * Set D0 = D1-2
811               *
812
813 07171 130  RUN100  D0=A                  D1 before SCOPCK call
814 07174 181          D0=D0- 2
815 07177 6F30         GOTO   RUN140         go RUN / CONT
816               *
817               * Label Error
818               *
819 0717B 8D00 =RUNER1 GOVLNG =STMTNF        Statement not found
          000
820               *
821               * <label>
822               *   If bad label or not in program scope
823               *      Error exit ---> "Stmt Not Found"
824               *   Collaspe Math stack incase of expression evaluation
825               *
826 07182 79A1 RUN110  GOSUB  LBLRCK         Check if Label Reference
827 07186 54F          GONC   RUNER1         Bad label reference
828 07189 878          ?ST=1  8              Not a label refernce ?
829 0718C 51           GOYES  RUN120
830 0718E AF8          B=A    W
831 07191 71F5         GOSUB  FINDLB         Find label across program
832 07195 45E          GOC    RUNER1         Label not found
833 07198 8E00         GOSUBL =COLLAP        Collapse Math stack
          00
834 0719E 581          GONC   RUN140         B.E.T.
835               *
836               * RUN | CONT with No parameters
837               *
838 071A1 74F7 RUN120  GOSUB  NULLP           Null Program ?
839 071A5 560          GONC   RUN125         no
840 071A8 6BA0         GOTO   MLPEXT         Return to Main Loop
841               *
842               * If CONT
843               *    If Continue Address # 0
844               *       Set Run Address @ CNTADR
845               * Position to first line of program
846               *
847 071AC 7940 RUN125  GOSUB  CNTCHK         CONT and non-zero Continue address
848 071B0 5B0          GONC   RUN145         go Run @ CNTADR
849 071B3 7F65         GOSUB  GETSTC         Get first line of program
```

```
                        *
                        * DO @ Start of Statement to Execute
                        *
                        * If CONT
                        *    Preserve DO
                        *    Collaspe Statement Buffer
                        *      Avoids subsequent "Return to Keyboard" to return
                        *      within new statement buffer containing "CONT..."
                        *
071B7 86A   RUN140 ?ST=0  sCONT          not Continue ?
071BA 21           GOYES  RUN150
071BC 136   RUN145 CDOEX                 Save DO on stack
071BF 06           RSTK=C
071C1 8E00         GOSUBL =STMBCL        Collapse Statement Buffer
      00
071C7 07           C=RSTK                Restore DO
071C9 134          DO=C
                        *
                        * RUN from program check
                        *   If Program running
                        *      Reenter BASIC loop @ RUNRT1
                        *      so Exceptions are checked
                        *   Prevents 10 RUN infinite loop
                        *
071CC 86D   RUN150 ?ST=0  PgmRun         Program not running ?
071CF 60           GOYES  RUN155
071D1 6513         GOTO   RUNRT1         Check exceptions before RUNning
                        *
                        * Entry point for SST
                        *   NoCont (S14) flag set
                        *
                        * Clear RUN Binary flag
                        * Clear SUSP Annunciator/Flag; PRGM Annunciator/Flag
                        *
071D5 844   RUN155 ST=0   sRUNBn         Clear RUN Binary flag
071D8 7371  RUN160 GOSUB  SFGPGM         Set PgmRun, PRGM Annunciator
                        *
                        * If Binary run
                        *    R2 = Binary File type
                        *    Poll for RUN of non BASIC file
                        *    Jump  to Binary file start address
                        *
071DC 864           ?ST=0  sRUNBn        Not RUN Binary ??
071DF 61            GOYES  RUN170
071E1 7305          GOSUB  Poll
071E5 13            CON(2) =pRUNnB       RUN with non BASIC file
071E7 560           GONC   RUN165        Memory Error ?
071EA 699D          GOTO   RUNERR        (BSERR)
071EE 136   RUN165 CDOEX                 Start of binary code
071F1 06           RSTK=C
071F3 01           RTN                   Jump to binary code
                        *
                        * Go RUN BASIC from DO
                        *  NoCont could be set from SST entry @ RUN150
                        *
```

```
    904 071F5 6442  RUN170 GOTO   BSCEX2         go RUN
```

```
905                     STITLE CNTCHK - Check for CONT and zero CONT address
906                ****************************************************************
907                ****************************************************************
908                **
909                ** Name:    CNTCHK  -  Check CONT & non-zero CONT addr
910                ** Name:    CHTCK2  -  Check CONT & non-zero CONT addr
911                **
912                ** Category:   LOCAL
913                **
914                ** Purpose:
915                **      Check if CONT and if Continue address is non-zero
916                **      Check if non-zero Continue address
917                **
918                ** Entry:
919                **      CNTCHK:
920                **        sCONT  =  1  if CONT command
921                **
922                ** Exit:
923                **      Carry Clear
924                **       C(A) = Continue Address
925                **       D0   = Continue address
926                **      CNTCHK: CONT cmnd and Continue address is non-zero
927                **      CNTCK2: Continue address is non-zero
928                **
929                **      Carry Set
930                **
931                **      CNTCHK: Either NOT CONT command / Continue addr = 0
932                **      CNTCK2: Continue Address is zero
933                **
934                ** Calls:      None
935                **
936                ** Uses.......
937                **  Exclusive: C(A),D0
938                **  Inclusive: C(A),D0
939                **
940                ** Stk lvls:   0
941                **
942                ** Detail:
943                **      This subroutine is necessary to determine if the
944                **      current program/subprogram is suspended and a
945                **      CONTinue is requested.
946                **
947                **      The current PRGMST and PRGMEN setting must remain, due to
948                **      subprograms.
949                **
950                **      A non-zero continue address also indicates where to resum
951                **      execution
952                **
953                **      SST uses CHKCK2
954                ** History:
955                **
956                **    Date     Programmer             Modification
957                ** --------   ----------   ----------------------------------
958                ** 12/16/82   JP           Added routine
959                **                         Happy Birthday Carone and Harms !
```

```
960                **
961                **********************************************************************
962                **********************************************************************
963 071F9 86A   CNTCHK ?ST=0  sCONT          Not CONT ?
964 071FC 00           RTNYES
965 071FE 1BE7 =CNTCK2 D0=(5) =CNTADR
          6F2
966 07205 146          C=DAT0 A             Read Continue Address
967 07208 134          D0=C                 Set D0 = Continue Address
968 0720B CE           C=C-1  A             Check for zero Continue address
969 0720D E6           C=C+1  A
970 0720F 01           RTN                  Return w/ carry set if zero address
```

```
 971                    STITLE SAVPCr - Save PC - Relative
 972                ******************************************************************
 973                ******************************************************************
 974                **
 975                ** Name:    SAVPCr  -  Save Prgm Counter - Relative
 976                **
 977                ** Category:   EXCUTL
 978                **
 979                ** Purpose:
 980                **      Save current D0 as relative position from PCADDR
 981                **      Useful if Memory Moves between D0 save & restore
 982                **
 983                ** Entry:
 984                **      D0 = Current Program Counter
 985                **
 986                ** Exit:
 987                **      Carry Clear
 988                **      S-R0-1 = Relative PC position from PCADDR
 989                **        C(A) = Relative PC position from PCADDR
 990                **        A(A) = (PCADDR)
 991                **          D0 @ S-R0-1
 992                **
 993                ** Calls:      D0=PCA
 994                **
 995                ** Uses.......
 996                **  Exclusive: A(A),C(A),D0,S-R0-1
 997                **  Inclusive: A(A),C(A),D0,S-R0-1
 998                **
 999                ** Stk lvls:   1
1000                **
1001                ** History:
1002                **
1003                **    Date     Programmer             Modification
1004                **  --------   ----------   ------------------------------------
1005                **  07/15/82   JP           Wrote routine
1006                **  03/17/83   JP           Packed in call to D0=PCA
1007                **  03/23/83   JP           Use S-R0-1 instead of STMTR0+64
1008                **
1009                ******************************************************************
1010                ******************************************************************
1011 07211 136    SAVPCr CDOEX                  C <-- Current PC
1012 07214 8E00           GOSUBL =D0=PCA        A <-- PCADDR
           00
1013 0721A E2             C=C-A  A              Relative offset from PCADDR
1014 0721C 1B67           D0=(5) =S-R0-1
           8F2
1015 07223 144            DAT0=C A              Save it
1016 07226 03             RTNCC
```

```
1017                    STITLE SST Execute
1018            ****************************************************************
1019            ****************************************************************
1020            **
1021            ** Name:    SST     -  Execute Single Step
1022            **
1023            ** Category:   STEXEC
1024            **
1025            ** Purpose:
1026            **      Execute Single Step Command
1027            **
1028            ** Entry:
1029            **      SST Key pressed
1030            **
1031            ** Exit:
1032            **      Through RUN150 to RUN Loop to Execute statement
1033            **      To MAINLP if Null Program
1034            **      To END30 if CNTADR @ End of Program
1035            **
1036            **      Errors:
1037            **       eFPROT  File protected
1038            **       eFTYPE  Illegal File type  (not BASIC - from GETSTe)
1039            **
1040            ** Calls:      AUTCLR,GETPRe,PRSCOP,NULLP,LDSST1,LDSST2,D1=TFB
1041            **             CPL#15,LIN#DC,CNTCHK,BLDDSP,DSPCNO,KEY?,DEBNCE,
1042            **             OUT2TC,CRLFOF,NOSCRL,LDCSET,DLFIBA,CURSFL,LDCOMP
1043            **
1044            ** Uses.......
1045            **  Exclusive: sSSTdc (S1),sSST (S2),TRFMBF (5 nibs)
1046            **  Inclusive: A-D,D0,D1,R0-R2,S0-S3,S5-S8
1047            **
1048            ** Stk lvls:   7
1049            **
1050            ** NOTE:
1051            **
1052            ** Detail:
1053            **      The statement line is displayed before executed
1054            **      Only the single statement to execute is displayed.
1055            **      The Line# precedes the statement.
1056            **      If part of a multi-statement line, an "@ " is
1057            **         displayed between the Line# and statement start.
1058            **      If followed by another statement, an "@ is displayed
1059            **
1060            **      For IF/THEN/ELSE
1061            **          The IF clause is displayed with the THEN clause
1062            **          If a multi-statement THEN
1063            **             The first statement is displayed
1064            **          If the <expression> is false & the ELSE is a branch
1065            **             The ELSE clause is never seen
1066            **
1067            **      Single stepping through:
1068            **
1069            **      10 LET A=B @ DISP A @ GOSUB "COMPUTE"   displays:
1070            **
1071            **      10 LET A=B @
```

```
1072          **      10@ DISP A @
1073          **      10@ GOSUB "COMPUTE"
1074          **
1075          ** NOTE:  The Cont Address to single step from will ALWAYS
1076          **        be past a trailing ELSE clause, thanks to the RUN Loop
1077          **
1078          ** Algorithm:
1079          **
1080          **      Send CR/LF                                    (CRLFOF)
1081          **      Clear AUTO Mode                               (AUTCLR)
1082          **      If private file                               (GETPRe)
1083          **         Error Exit
1084          **      If CNTADR = 0
1085          **         Get program scope                          (PRSCOP)
1086          **         Delete FIB,Collapse ASSIGN table           (DLFBIA)
1087          **         If Null Program                            (NULLP)
1088          **            Return - Do nothing                     (MAINLP)
1089          **      else
1090          **  1:     If at end of program: CNTADR >= PRGMEN-2
1091          **            Set SST @ End of program flag           (sSST)
1092          **            go Wait for SST key, then END prgm      (goto 20
1093          **      Set SST Decompile flag                        (sSSTdc)
1094          **      Set NoCont Flag for Run Loop                  (NoCont)
1095          **      Save RUN address @ TRFMBF                     (D1=TFB)
1096          **      If Start of line  (@ EOL)
1097          **         go Decompile first statement on line       (LDSST1)
1098          **      else
1099          **         Compute current line#                      (CPL#15)
1100          **         Set bounds for Decompile                   (LDCSET)
1101          **         Output line#                               (LIN#DC)
1102          **         Output "@ "                                (OUT2TC)
1103          **         Restore D1 <-- Start of statement
1104          **         Decompile statement in multi-stmt line(LDSST2)
1105          **      Set D1 @ Start Input Buffer
1106          **      Send characters to Display Buffer             (SNDCHO)
1107          **      Set No scroll needed at this time             (NOSCRL)
1108          **      Send Cursor Far Left to see 1st 22 chars      (CURSFL)
1109          **      Build Display                                 (BLDDSP)
1110          **      Clear SST @ End of program flag               (sSST)
1111          **  2:  If no keys in buffer                          (KEY?)
1112          **         Wait for SST Key to go up                  (DEBNCE)
1113          **      Send CR/LF with no delay                      (CRLFOF)
1114          **      If SST @ End of program                       (sSST)
1115          **         go END program                             (END25)
1116          **      else
1117          **         Restore Execution address                  (D1=TFB)
1118          **         go RUN --- Execute statement               (RUN155)
1119          **
1120          ** History:
1121          **
1122          **     Date     Programmer          Modification
1123          **   --------   ----------   ------------------------------
1124          **   01/11/83   JP           Wait for SST Key before END prgm
1125          **   01/31/83   JP           Added CURSFL call before BLDDSP
1126          **   03/28/83   JP           Save Run addr @ TRFMBF cus of
```

```
1127                **                                stack levels with Decompile
1128                **   04/25/83   JP               Set NoCont after EOF check
1129                **   04/25/83   JP               GOTO RUN155, avoid Exception check
1130                **   04/25/83   JP               CLRST to distinquish from RUN
1131                **   05/17/83   JP               If SST@PRGMEN;check for ENDSUB/DEF
1132                **   05/17/83   JP               Set NoCont before EOF check
1133                **                                incase SST @ "implicit" ENDSUB
1134                **
1135                ****************************************************************
1136                ****************************************************************
1137 07228 71F0  =SST    GOSUB  CrLfof          Send CR/LF
1138 0722C 8E00         GOSUBL =AUTCLR         Clear AUTO Mode
           00
1139 07232 7635         GOSUB  GETPRe          Check Protection, Error if so
1140                *
1141                * Set NoCont to stop execution after this statement execute
1142                * If Continue Address = 0
1143                *    Get and update Program Scope
1144                *    Delete FIB, collapse ASSIGN table
1145                *    If Null Program
1146                *       Return - Do nothing
1147                *
1148 07236 85E   SST010 ST=1   NoCont          Set NoCont to stop after stmt exec
1149 07239 71CF         GOSUB  CNTCK2          Continue address # 0 ?
1150 0723D 5C1          GONC   SST020          C = Continue address
1151 07240 8E94         GOSUBL PRSCOP          Get program scope
           90
1152 07246 8F00         GOSBVL =DLFIBA         Delete FIB, collapse ASSIGN table
           000
1153 0724D 7847         GOSUB  NULLP
1154 07251 512          GONC   SST025          Not Null program
1155 07254 8C00  MLPEXT GOLONG =MAINLP         Return / Do nothing
           00
1156                *
1157                * If CNTADR @ Program End
1158                *    Set SST flag for END entry
1159                *    go Wait for SST Key to come up, then END program
1160                *
1161 0725A 1F76  SST020 D1=(5) =PRGMEN
           5F2
1162 07261 143          A=DAT1 A               Program End past last EOF
1163 07264 CC           A=A-1  A               Position to last EOF
1164 07266 CC           A=A-1  A
1165 07268 8B6          ?C<A   A               Not at End of Program ?
1166 0726B B0           GOYES  SST030
1167 0726D 852          ST=1   sSST            Set SST flag @ End of program
1168 07270 527          GONC   SST055          Wait for SST Key; End program
1169                *
1170                * Save Start of statement
1171                *   Save @ TRFMBF because call to Line Decompile uses 6 levels
1172                * Set SST Decompile flag
1173                * Set D1 = Start of statement
1174                *
1175 07273 137   SST025 CD1EX                   First line of file
1176 07276 79A0  SST030 GOSUB  D1=TFB          Set D1 @ TRFMBF
```

```
1177 0727A 145          DAT1=C A            Save Run Address
1178 0727D 851          ST=1   sSSTdc       Set SST Decompile flag
1179 07280 135          D1=C
1180             *
1181             * If start of new line  (@ EOL)
1182             *    Decompile first statement on line
1183             *
1184 07283 14F          C=DAT1 B            Read EOL terminator
1185 07286 90E          ?C#0   P            Not EOL ?
1186 07289 F0           GOYES  SST040
1187 0728B 171          D1=D1+ 2            Position to Line#
1188 0728E 8E00         GOSUBL =LDSST1      Decompile first statement
           00
1189 07294 6230         GOTO   SST050
1190             *
1191             * Multi-Statement Line
1192             *    Compute & Output Line#
1193             *    Set bounds for Decompile
1194             *    Output "@ "
1195             *
1196 07298 137   SST040 CD1EX               C <-- Start of statement
1197 0729B 7BE5         GOSUB  =CPL#15      Compute Line#
1198 0729F 8E00         GOSUBL =LDCSET      Set bounds for Decompile
           00
1199 072A5 8E00         GOSUBL =LIN#DC      Decompile Line#
           00
1200 072AB 3304         LCASC  \ @\
           02
1201 072B1 8E00         GOSUBL =OUT2TC      Output "@ "
           00
1202             *
1203             *    Set D1 <-- Start of statement
1204             *    Decompile statement past "@"
1205             *
1206 072B7 7860         GOSUB  D1=TFB       Set D1 @ TRFMBF, A<--address
1207 072BB 131          D1=A                D1 <---Run Address
1208 072BE 171          D1=D1+ 2            Step over @ | EOL
1209 072C1 8E00         GOSUBL =LDSST2      Decompile statement
           00
1210             *
1211             * Statement is Decompiled
1212             *    On RETURN from Decompile: B(A) = #chars to send
1213             * Send characters to Display Buffer from Output Buffer
1214             * Set no scroll needed
1215             * Send Cursor Far Left to Display first 22 chars
1216             * Build display
1217             * Clear SST @ End of program flag
1218             *
1219 072C7 8E00  SST050 GOSUBL =DSPCNO      Send chars to display from OUTBS
           00
1220 072CD 8F00         GOSBVL =NOSCRL      No scroll needed
           000
1221 072D4 8E00         GOSUBL =cursfl      Send Cursor Far Left
           00
1222 072DA 8E00         GOSUBL =BLDDSP      Build display buffer
```

```
          00
1223 072E0 842          ST=0   sSST          Clear SST @ End of program
1224               *
1225               * If no keys in buffer
1226               *    Wait for SST key to go up before Executing statement
1227               *    On return from DEBNCE:
1228               *       D0 @ DISINT
1229               *       SST Key in KCOL4 (Key Column 5)
1230               *
1231 072E3 8E00  SST055 GOSUBL =KEY?          Keys in buffer ?
          00
1232 072E9 591          GONC   SST070         Yes - go Execute
1233 072EC 26    SST060 P=     6              6/512 wait before Key Scan
1234 072EE 8E00         GOSUBL =DEBNCE        Debounce, Scan keys
          00
1235 072F4 184          D0=D0- 5              Pos to Key Column 5 (KCOL4)
1236 072F7 14A          A=DAT0 B              SST Key in bit 0 of A(1)
1237 072FA 81C          ASRB                  Shift SST Key to bit 3 of A(0)
1238 072FD A04          A=A+A  P              SST Key still down ?
1239 07300 4BE          GOC    SST060         Yes
1240               *
1241               * Send CR/LF with Cursor ON
1242               * If SST @ End of program
1243               *    go END program --- checking for ENDSUB/ENDDEF
1244               * else
1245               *    Restore Statement Execution Address
1246               *    go RUN ---- Execute statement, then PAUSE
1247               *           ---- Avoid exception checking
1248               *
1249 07303 7610  SST070 GOSUB  CrLfof         Send CR/LF with no delay
1250 07307 862          ?ST=0  sSST           Not SST @ End of program ?
1251 0730A 60           GOYES  SST080
1252 0730C 6E33         GOTO   END10          Go end program
1253 07310 7F00  SST080 GOSUB  D1=TFB         D1 <-- Saved run addr, A <-- addres
1254 07314 130          D0=A                  Set D0 @ statement start/run addr
1255 07317 08           CLRST                 Clrst to distinquish from CONT
1256 07319 6BBE         GOTO   RUN155         go execute it
1257 0731D 8C00  CrLfof GOLONG =CRLFOF        Send CR LF
          00
1258               *
1259               * Set D1 @ Save area for Run address  (TRFMBF)
1260               * Read Run address into A
1261               *
1262 07323 1F5C  D1=TFB D1=(5) =TRFMBF        Save area for Run address
          8F2
1263 0732A 143          A=DAT1 A              Read run address
1264 0732D 01           RTN
```

```
1265                    STITLE Label Reference Check LBLRCK
1266                ********************************************************************
1267                ********************************************************************
1268                **
1269                ** Name:    LBLRCK  -  Check if valid label reference
1270                ** Name:    FILXQT  -  Check if valid label reference
1271                **
1272                ** Category:   LOCAL
1273                **
1274                ** Purpose:
1275                **      Check if Valid Label Reference
1276                **
1277                ** Entry:
1278                **   LBLRCK:
1279                **      A = Possible Label Reference Token
1280                **      D0  past Label Reference Token
1281                **   FILXQT
1282                **      D0 @ Label within statement
1283                **
1284                ** Exit:
1285                **      Carry Clear
1286                **         Bad Label reference
1287                **
1288                **      Carry Set
1289                **       FILXQT
1290                **         Good label name
1291                **       LBLRCK:
1292                **         S8=0   Simple label name
1293                **                A = Label name
1294                **         S8=1   Not a Label Reference
1295                **
1296                ** Calls:      FILXQ^
1297                **
1298                ** Uses.......
1299                **  Exclusive:  A,C,S8
1300                **  Inclusive:  A,C,S8,S1,S2,S7,A-D,D0,D1,S-R1-0 thru S-R1-3,
1301                **              STMTD0,R0-R4, all of Function scratch
1302                **
1303                ** Stk lvls:   6
1304                **
1305                ********************************************************************
1306                ********************************************************************
1307 0732F 858   LBLRCK ST=1   8              Set S8 for Not a Label Ref
1308 07332 3100         LC(2)  =tLBLRF
1309 07336 966          ?A#C   B              Not a Label Reference ?
1310 07339 00           RTNYES
1311 0733B 8E00  FILXQT GOSUBL =FILXQ^        Get label into A
          00
1312 07341 500          RTNNC                 Illegal label - Carry clear
1313 07344 B47          D=D+1  S              Device following name =D(S)=F
1314 07347 500          RTNNC                 Not a simple filename/label
1315 0734A 848          ST=0   8
1316 0734D 01           RTN                   Carry set/Simple filename
1317                *
1318                * Clear SUSP annunciator
```

```
1319            * Set PgmRun flag
1320            * Set PRGM Annunciator
1321            *
1322 0734F 7E36 =SFGPGM GOSUB  CLSUSP          Clear SUSP annunciator
1323 07353 85D          ST=1   PgmRun          Set Program Running flag
1324 07356 312C         LC(2)  =f1PRGM         Turn on PRGM Annunciator
1325 0735A 8C00  SFLAGs GOLONG =sflags
          00
1326            *
1327            *  Update CNTADR from C
1328            *  Move Continue Address to D0
1329            *    Increment by 2 to point into "next" statement
1330            *  Check if CNTADR + 2 still within Program Scope
1331            *    Checks for  Continue Address past END of program
1332            *  If CNTADR+2 >= PRGMEN
1333            *     Decrement pointer back into current statement
1334            *     This causes CURRL to be computed on current statement
1335            *  else
1336            *     CURRL will be computed on Continue statement
1337            *
1338            *  Set SUSP annunciator
1339            *
1340            *  SETSU1:  KBRTCK entry:  D0 @ CNTADR
1341            *
1342            *
1343 07360 1BE7 =SETSUS D0=(5) =CNTADR         Update CNTADR
          6F2
1344 07367 144          DAT0=C A
1345 0736A 134          D0=C                  Save CNTADR
1346 0736D 161  =SETSU1 D0=D0+ 2              Move D0 into line
1347 07370 132          ADOEX                 Move D0 to A
1348 07373 7B43         GOSUB  Scopck
1349 07377 130          D0=A                  Restore D0
1350 0737A 550          GONC   SETSU2         Within program scope
1351 0737D 181          D0=D0- 2              Move back into current statemtn
1352 07380 311C  SETSU2 LC(2)  =f1SUSP
1353 07384 65DF         GOTO   SFLAGs         Set SUSP annunciator
1354            *
1355            * Update Continue address
1356            * Set SUSP annunciator
1357            * Compute line# containing Continue Address
1358            * Update CURRL
1359            *
1360 07388 74DF =CNTCUR GOSUB  SETSUS         Update CNTADR, Set SUSP
1361 0738C 77F4         GOSUB  CPL#10         Compute line#
1362 07390 8D00 =savel# GOVLNG =SAVEL#        Update CURRL
          000
```

```
1363                    STITLE FETCH Execute
1364            ********************************************************************
1365            ********************************************************************
1366            **
1367            ** Name:    FETCH   -  Statement Execute
1368            **
1369            ** Category:   STEXEC
1370            **
1371            ** Purpose:
1372            **      Execute FETCH statement
1373            **
1374            ** Entry:
1375            **      D0 past FETCH token
1376            **
1377            ** Exit:
1378            **      Through FTCHKY if FETCH KEY
1379            **      Through DCPLIN if FETCH
1380            **      Through AUTLN2 if Line# not found
1381            **
1382            **      Errors
1383            **        eFTYPE   Not BASIC file
1384            **        eFPROT   Private file
1385            **        eSTMNF   Stmt  not found
1386            **
1387            ** Calls:      GETPRe,FINDLR,FCHLBL,LBLRCK,NULLP,CURRLO
1388            **
1389            ** Uses.......
1390            **  Exclusive:     A-D,S4,S8,R3,CURRL,R1,D0,D1,R2
1391            **  Inclusive:     A-D,S0,S1,S2,S4,S7,S8,R0-R3,CURRL,STMTD0
1392            **                 STMTR1 (all of it), All of function scratch
1393            **
1394            **  R1 = Saved D0
1395            **
1396            ** Stk lvls:   7
1397            **
1398            ** Detail:
1399            **      FETCH KEY ....
1400            **      FETCH [ <lineno> | <label> ]
1401            **
1402            ** Algorithm:
1403            **
1404            **      If next token = KEY
1405            **         go Execute FETCH KEY                  (FTCHKY)
1406            **
1407            **      Save D0                                  (R1)
1408            **      Read current header; Error if not BASIC  (GETPRe)
1409            **      If private file --> Error
1410            **      Restore D0
1411            **      Read and skip next token
1412            **      If line#
1413            **         Skip jump address & Read line#
1414            ** 0:      Find line#                            (FINDLR)
1415            **         If line# found
1416            ** 1:         go Decompile and Display @ D1      (DCPLIN)
1417            **         else
```

```
1418            ** 2:         Update CURRL <-- Line# not found
1419            **            go Display "line#" w/ null line    (AUTLN2)
1420            **
1421            **      Check if Label Reference                 (LBLRCK)
1422            **      If label reference
1423            **         If simple label
1424            **            Find label across file             (FCHLBL)
1425            **            If found
1426            **               go Decompile & display line     (goto 1)
1427            **            else
1428            **         else
1429            **            Error Exit --> Label not found
1430            **      else  (No paramters)
1431            **         If CURRL # 0
1432            **            Line# <-- CURRL
1433            **            go Find Line#                      (goto 0)
1434            **         else
1435            **            If NULL program                    (NULLP)
1436            **               Display "10" w/ Null line       (goto 2)
1437            **            else
1438            **               go Display first line of file   (CUR020)
1439            **
1440            ** History:
1441            **
1442            ** Date        Name          Modification
1443            ** --------    -----------   --------------------------------
1444            ** 04/12/83    JP            If CURRL=0 and not NULLP
1445            **                              Display first line of file
1446            **
1447            ********************************************************************
1448            ********************************************************************
1449 07397 0000          REL(5) =FETCHP        Parse Address
         0
1450 0739C 3100 =FETCH  LC(2)  =tKEY
1451 073A0 14A          A=DAT0 B
1452 073A3 966          ?A#C   B              not FETCH KEY ?
1453 073A6 C0           GOYES  FTC010
1454 073A8 161          D0=D0+ 2              Skip KEY token
1455 073AB 8D00         GOVLNG =FTCHKY        Execute FETCH KEY
         000
1456            *
1457            * If private file
1458            *    Error through GETPeF
1459            *
1460 073B2 136   FTC010 CD0EX
1461 073B5 109          R1=C                  Save D0 (Program counter)
1462 073B8 136          CD0EX
1463            *
1464            * Error exits if Private or Non BASIC
1465            *
1466 073BB 7DA3         GOSUB  GETPRe         Get protection and filetype
1467            *
1468            * Read & skip token
1469            *
1470 073BF 119   FTC020 C=R1                  Restore D0
```

```
1471 073C2 134          DO=C
1472 073C5 14A          A=DAT0 B              Read token
1473 073C8 161          DO=DO+ 2              Skip token
1474 073CB 3100         LC(2)  =tLINE#
1475 073CF 966          ?A#C   B              not Line#
1476 073D2 72           GOYES  FTC050
1477            *
1478            * FETCH <line#>
1479            *
1480 073D4 164          DO=DO+ 5              Skip Jump address
1481 073D7 8F00  FTC025 GOSBVL =FINDLR        Read & Find line# across file
          000
1482 073DE 501          GONC   FTC040         Line# not found
1483            *
1484            * Decompile and Display Line @ D1
1485            *
1486 073E1 854   FTC030 ST=1   4              External Entry flag
1487 073E4 8D00 =dcplin GOVLNG =DCPLIN        Decompile and display
          000
1488            *
1489            * Line# not found
1490            *    Update CURRL
1491            *    go Display Line# with Null line
1492            *
1493            * AUTLN2 entry:
1494            *        B(A) = Line# to display
1495            *
1496            * On return from NULLP
1497            *  C(A) = 00011 (oBSsod has been loaded)
1498            * FETCH needs 00010
1499            *
1500 073EB CE    FTC035 C=C-1  A
1501 073ED D5           B=C    A
1502
1503 073EF 8C00  FTC040 GOLONG =AUTLN2
          00
1504            *
1505            * Error Exit
1506            *
1507 073F5 658D  FTCER2 GOTO   RUNER1         Label not found
1508            *
1509            * Check if Label Reference
1510            *
1511 073F9 723F  FTC050 GOSUB  LBLRCK         Label Reference Check
1512 073FD 57F          GONC   FTCER2         Bad Label name
1513 07400 878          ?ST=1  8              Not label reference ?
1514 07403 C1           GOYES  FTC060         FETCH w/o Parms
1515 07405 103          R3=A                  Label name
1516 07408 7024         GOSUB  FCHLBL         Find label across file
1517 0740C 48E          GOC    FTCER2         Label not found
1518 0740F 51D          GONC   FTC030         B.E.T. Decompile @ D1
1519            *
1520            * Cursor key hit and program is null
1521            *
1522 07412 8E00 =CURSNP GOSUBL =AUTOCK
```

```
            00
1523 07418 560          GONC   FTC060         If in AUTO mode, disp currl
1524 0741B 683E         GOTO   MLPEXT          Else goto MAINLP
1525              *
1526              * FETCH with No Parameters
1527              *    If current line # 0
1528              *       go Find current line
1529              *    else
1530              *      If Null program
1531              *         go Display Line# 10 with a null line
1532              *      else
1533              *         go Display first line of file
1534              *         Enter CURTOP code below CR/LF
1535              *           Assumes sCURBOT=0 (S3)
1536              *
1537              *         CURRL is zeroed IF
1538              *           Return to keyboard
1539              *           Error on nonBASIC file
1540              *           Computing line# and PCADDR not in program scope
1541              *             Caused from CALL/GOSUB from keyboard and
1542              *             ATTN key before first line executed
1543              *
1544 0741F 8F00  FTC060 GOSBVL =CURRLO        Current Line# = 0 ?
            000
1545 07426 408          GOC    FTC025         No - go Find current line
1546 07429 7C65         GOSUB  NULLP          Null program ?
1547 0742D 4DB          GOC    FTC035         Yes -
1548 07430 8D00         GOVLNG =CUR020        Display first line of file
            000
```

```
1549                    STITLE BASIC Interpreter Loop
1550            ********************************************************************
1551            ********************************************************************
1552            **
1553            ** Name:(S) BSCEXC  -  BASIC Stmt/Pgm Execution: Keyboard Exec
1554            ** Name:(S) BSCEX2  -  BASIC Stmt/Pgm Execution: Program Exec
1555            ** Name:(S) BSCEXT  -  BASIC Stmt/Pgm Exec: Reentry into BASIC lo
1556            ** Name:(S) RUNRT1  -  Stmt reentry to BASIC loop;sERROR,sENDx cl
1557            ** Name:(S) RUNRTN  -  Stmt reentry to BASIC loop; sERROR cleared
1558            ** Name:(S) ERRRTN  -  Error Exit reentry to BASIC loop
1559            **
1560            ** Category:   SYSTEM
1561            **
1562            ** Purpose:
1563            **      BASIC interpreter loop for program/statement execution
1564            **      Complete execution of RUN | CONT command
1565            **
1566            ** Entry:
1567            **      BSCEXC: NoCont flag clear
1568            **                Keyboard Execute entry: PgmRun = 0
1569            **      BSCEX2: If PgmRun = 1
1570            **                Program to be executed
1571            **                DO @ EOL of prior statement
1572            **                (RUN,CONT,SST entry)
1573            **                (If NoCont=1 then SST)
1574            **              If PgmRun = 0
1575            **                Statement to be executed
1576            **                DO @ Statement length byte of statement
1577            **              Polls on entering BASIC interpreter
1578            **
1579            **      BSCXLP: LABEL entry if within Multi-statement line
1580            **                DO @ EOL or @ of next statement to execute
1581            **
1582            **      BSCEXT:
1583            **              Return to Keyboard "Reentry"
1584            **              RUN/CALL Binary return from "ENDBIN"
1585            **              ENDALL from PURGE/MERGE current file
1586            **                Filetype is read
1587            **                If BASIC and Program running (S13=1)
1588            **                   DO @ Next stmt to execute
1589            **                   SUSP will occur
1590            **                   Exceptions are NOT checked
1591            **                sERROR=1  (S0) --> Error has occured
1592            **                If not an error, Flush print buffers
1593            **                Poll on exiting BASIC interpreter
1594            **                Clears flags, goto Main Loop
1595            **
1596            **      RUNRT1: Statement reentry into BASIC loop
1597            **              sENDx, sERRORx cleared
1598            **      RUNRTN: Statement reentry into BASIC loop
1599            **              sERRORx cleared --- used by END stmt
1600            **      ERRRTN: Error exit reentry into BASIC loop
1601            **              Assemes sERROR set; sENDx clear
1602            **
1603            ** Exit:
```

```
1604            **      Jump to individual execute routine for statement
1605            **      S0-S11 are cleared before jumping
1606            **
1607            **      ALL statements MUST return through NXTSTM or
1608            **        directly to RUNRT1/RUNRTN with D0 set properly
1609            **
1610            **        CALL, END SUB, FN, GOTO, GOSUB jump to RUNRT1
1611            **        NXTSTM returns to RUNRTN
1612            **        Errors return to ERRRTN
1613            **        SST @ Program End returns to BSCEXT
1614            **        Binaries return to BSCEXT (from ENDBIN)
1615            **
1616            **        RUNRT1:  Clears sENDx flag, indicating not END stmt
1617            **                 Clears sERROR
1618            **        RUNRTN:  Assumes sENDx flag set appropriately
1619            **                 Clears sERROR
1620            **
1621            **      If continuing execution:
1622            **
1623            **      Timers are serviced at the end of each stmt execute
1624            **        If a timer expires & is within current program scope
1625            **           The appropriate ON TIMER code is jumped to.
1626            **           Statement execution will return to RUNRTN
1627            **
1628            **      Execution stops if:
1629            **        End of program reached | STOP/END statement in program
1630            **        End of line of calculator statement
1631            **        Don't Continue (NoCont) flag set from:
1632            **          PAUSE, ATTN, Error Message Routine
1633            **          END/STOP within Program
1634            **          End of Program reached
1635            **          SST
1636            **          END(SUB),END(DEF), RETURN from keyboard
1637            **        Error flag (sERROR) set from Error Message routine
1638            **
1639            **
1640            ** Calls:      EXCADR,CK"ON",BASCHK,SFLGCP,FLUSHA,CNTCUR,CKSREQ,
1641            **             EOLSCN,USRSTA,GTMRA+,FPOLL,ALMSRV,SCOPCK,TRFCK-,
1642            **             UPDPC,RDCHD+
1643            **
1644            ** Uses.......
1645            **  Exclusive: A,C,D1,D0,S13,S14,PCADDR,CURRL,R0,S0,S1
1646            **  Inclusive: A-D,D1,D0,S13,S14,PCADDR,CURRL,R0,S0-S7,
1647            **             SCRTCH (32 nibs),flPRGM,flSUSP,ANNAD1-4,STMTD1
1648            **
1649            **           PCADDR must not be used for anything else
1650            **           sENDx  = END/STOP  Statement        S1
1651            **                    NXTSTM explicitly clears
1652            **                    RUNRT1 explicitly clears
1653            **           sERROR = ERROR occured              S0
1654            **                    RUNRTN,RUNRT1 clear
1655            **                    MFERR/BSERR sets
1656            **           Except = Service Request            S12
1657            **           PgmRun = Running program            S13
1658            **           NoCont = Don't Continue Execution   S14
```

```
1659            **          Trace  = TRACE Mode                    S15
1660            **
1661            ** Stk lvls:   >=4
1662            **
1663            ** Algorithm:
1664            **
1665            ** BSCEXC:  Clear No Continue of Program flag      (NoCont)
1666            ** BSCEX2:  Place current DO into RO
1667            **          Fast poll on entering BASIC interpreter(pBSCen)
1668            **          If not running                        (not PgmRun)
1669            **             go update PC address               (goto BSCX+)
1670            ** BSCXLP:  Read & Move past EOL | @
1671            **          If EOL and not running
1672            **             go Read filetype then              (goto BSCEXT)
1673            **             go exit BASIC                      (goto BSCEX+)
1674            **          If @  (multi-statement line)
1675            **             go Update PC address               (goto BSCX+)
1676            **          If End of current program             (PRGMEN > D0)
1677            **             go execute END statement                (END10)
1678            **          Skip line#
1679            **   BSCX+: Update PCADDR @ stmt  length byte       (PCADDR)
1680            **          Save addr @ statement length byte      (PCADDR)
1681            **          Skip statement length byte
1682            **          Clear lower status                     (S0-S11)
1683            **          Read Begin BASIC token
1684            **          If not Begin BASIC token range         (BASICs)
1685            **             Call Assignment Execute             (ASNMNT)
1686            **             Skip to next statement              (NXTSTM)
1687            **          else
1688            **             Move past BASIC token
1689            **             Calculate Execution addr            (EXCADR)
1690            **             Jump to Execution routine
1691            **
1692            ** Statement Execute Return: (from NXTSTM or directly)
1693            **
1694            **  RUNRT1: Clear END execute flag                  (sENDx)
1695            **  RUNRTN: Clear ERROR flag                       (sERROR)
1696            **  ERRRTN: Collaspe Math Stack
1697            **          If ERROR                               (sERROR)
1698            **             Skip exception checking             (goto 6)
1699            **          If no exceptions                      (Except=0)
1700            **             If no hardware service request       (SREQ)
1701            **                If any pending alarm set        (PNDALM)
1702            **                   Save D0 on stack
1703            **                   go Process timers           (goto 3)
1704            **                go continue                    (goto 6)
1705            **          Save D0 on stack
1706            **          Check Service requests                 (CKSREQ)
1707            **          If no exceptions                      (Except=0)
1708            **             go Restore D0 and continue          (goto 5)
1709            **          Clear Exception Flag                   (Except)
1710            **          Fast Poll on Exception                 (pExcpt)
1711            **          Restore low status from DSPSTA         (USRSTA)
1712            **       3: If ATTN Key hit                          (CKON)
1713            **             Set NoCont flag                       (S14)
```

```
1714          **          else
1715          **             If Program running
1716          **                Load mask to check Timer bits
1717          **                Read Pending Alarm field          (PNDALM)
1718          **       4:       If Timer expired     (Bit 0|1|2 of PNDALM)
1719          **                   Get Timer Address              (GTMRA+)
1720          **                   If non-zero Timer address
1721          **                      Verify address in prgm scope(SCOPCK)
1722          **                   If within scope           (Carry clear)
1723          **                      Clear timer bit in PNDALM
1724          **                      Set Except if anthr timer due(ALMSRV)
1725          **                      If TRACING                  (TRFCK-)
1726          **                         Update PCADDR @ next stmt to exec
1727          **                      C <-- ON TIMER a dress
1728          **                      Set ONTIMER statement flag (sONTMR)
1729          **                      Clr ONERROR statement flag (sONERR)
1730          **                      go process ON TIMER stmt   (ONTIMR)
1731          **                go Check if any other Timers off (goto 4)
1732          **       5: Restore DO
1733          **          Clear Error occured                     (sERROR)
1734          **       6: If Continue                          (not NoCont)
1735          **             go process next of statement         (BSCXLP)
1736          **          else                                    (NoCont)
1737          ** BSCEXT:     Clear PRGM Annunciator               (SflgCp)
1738          **             Read Filetype                        (RDCHD+)
1739          **             If non-BASIC file                    (BASCHK)
1740          **                go exit BASIC                 (goto BSCEX+)
1741          **             If not running                    (not PgmRun)
1742          **                go exit BASIC                 (goto BSCEX+)
1743          **             else
1744          **                If not END/STOP execute            (sENDx)
1745          **                   If ELSE
1746          **                      Skip to End of Line        (EOLSCN)
1747          **                   Update Continue Address
1748          **                   Set SUSP Annunc/Flag          (SFLAGS)
1749          **                   Compute & Update current line (CNTCUR)
1750          ** BSCEX+:
1751          **          If not an error                         (sERROR)
1752          **             Flush all buffers                    (FLUSHA)
1753          **          Fast Poll on Exiting BASIC interp       (pBSCex)
1754          **          Clear Don't Continue flag               (NoCont)
1755          **          golong MAIN Loop                        (MAINLP)
1756          **
1757          ** A note on CNTADR and CURRL:
1758          **
1759          ** When execution is not continued:
1760          **    The current Line# is updated
1761          **    If not an END/STOP statement and BASIC file
1762          **       The Continue Address is updated to the next statement
1763          **
1764          ** If the end of program scope (@ PRGMEN) | END/STOP
1765          **    CNTADR = 0
1766          **    Continue Address is NOT updated at end of BSC Loop
1767          **    Current Line is not touched
1768          **    This is normal program execution termination.
```

```
1769            **    CONT will start execution at the start of the prog
1770            **
1771            ** If the end of program scope is NOT reached, but exec stops:
1772            **    CNTADR = Current DO
1773            **    CURRL  = Line# of Continue address
1774            **    Current line always points @ CNTADR statement
1775            **
1776            **    For Error Messages
1777            **        CNTADR = Statement in error
1778            **        CURRL  = Line# of error
1779            **
1780            **    For ATTN Key
1781            **        CNTADR = Next statement to execute
1782            **        CURRL  = Line# containing next statement to execute
1783            **
1784            **    For PAUSE:
1785            **        CNTADR = Statement after PAUSE
1786            **        CURRL  = Line# containing statement after PAUSE
1787            **
1788            **
1789            ** History:
1790            **
1791            **   Date      Programmer     Modification
1792            ** --------    ----------     --------------------------------
1793            ** 02/04/83    JP             Added ALMSRV call if Timer due
1794            ** 03/07/83    JP             Packed: added UPDCRL call
1795            ** 03/08/83    JP             Clear sEXTGS before ONTIMR jump
1796            ** 03/28/83    JP             If ERROR, skip exception check,sERR
1797            ** 03/28/83    JP             If not error, flush buffers
1798            ** 04/04/83    JP             If tracing & ON TIMER update PCADDR
1799            ** 04/04/83    JP             Preserve S0-S11 during pExcept
1800            ** 04/08/83    JP             Zero Timer bit ONLY when servicing
1801            ** 04/08/83    JP             If no exceptions/SR check Timer bit
1802            ** 04/21/83    JP             Don't SUSP if non BASIC file
1803            ** 04/25/83    JP             Pass filetype in pBSCex poll ALWAYS
1804            ** 04/25/83    JP             Changed BSCEXT entry point
1805            ** 05/18/83    JP             Check Attn after pExcept (CK"ON")
1806            ** 06/17/83    JP             Update CURRL only if SUSPending
1807            ** 06/17/83    JP             CURRL points at CNTADR statement
1808            **
1809            ********************************************************************
1810            ********************************************************************
1811            ****************************************************************
1812            ****************************************************************
1813            **
1814            ** Name:(S) pBSCen  -  Poll entering BASIC interpreter
1815            **
1816            ** Category:   POLL
1817            **
1818            ** Type:       FPOLL
1819            **
1820            ** Purpose:
1821            **      Fast poll when entering BASIC interpreter
1822            **
1823            ** Should poll be "Handled" (return with XM=0)?:
```

```
1824          **      No - Either this poll is a TAKE OVER poll
1825          **           or it should continue to ALL LEX files
1826          **
1827          ** Meaning of "Handling" Poll (what does code do if handled?):
1828          **      Take over BASIC interpreting
1829          **      Set up information/buffers/flags before execution
1830          **      begins, then let Poll continue
1831          **
1832          ** Entry conditions for handler (registers, ST, RAM, etc.):
1833          **      --Carry set on entry iff fastpoll--
1834          **      B[A] = Poll number (pBSCen)
1835          **      HEX mode.
1836          **      P=0.
1837          **
1838          **      If PgmRun (S13)
1839          **         Program about to be executed (RUN/CONT/SST)
1840          **           If NoCont (S14)
1841          **              SST (Single stepping)
1842          **           If sCONT (S10)
1843          **              Continue
1844          **           If sCONTK (S9)
1845          **              CONT or RUN Key
1846          **         R0 @ EOL or "@" prior to statement to execute
1847          **      else
1848          **         Keyboard execution from Statement Buffer
1849          **         R0 @ Statement length byte of statement
1850          **
1851          ** Normal exit conditions from handler if handled (ST, RAM,
1852          ** registers, etc.):
1853          **      HEX mode.
1854          **      This poll should never be "normally" handled
1855          **      Either the LEX file takes over or allows other
1856          **      LEX files to respond.
1857          **
1858          ** Normal exit conditions from handler if not handled (ST, RAM,
1859          ** registers, etc.):
1860          **      HEX mode.
1861          **      XM=1.
1862          **      Global status intact
1863          **      Do not use S3
1864          **      R0 must be PRESERVED !!!!!
1865          **
1866          ** Available subroutine levels:
1867          **      --FPOLL handler is two levels deeper than caller--
1868          **      This is a "top level" poll --- 6 levels available--
1869          **      Must be able to return to Poll routine
1870          **
1871          ** NOTE:
1872          **   GOSUB,CALL,FNx invoked from the keyboard will appear
1873          **   as Keyboard Execute.  The PgmRun flag will be clear.
1874          **
1875          **   Program execution will begin with NO indication.
1876          **
1877          **   For CALL: pCALSV polls when CALL execute begins
1878          **       FNx:  pFNIN  polls when FNx executes begins
```

```
1879            **
1880            ** Binary Files will always be RUN/CONT from the start of
1881            **   the file... it is "impossible" to systematically
1882            **   return a meaningful CONTinue address through the BASIC
1883            **   loop.  If a Binary file wishes to implement CONT...
1884            **   it should respond to the pBSCex poll:
1885            **      If current filetype is Binary and PgmRun=1
1886            **         Update CNTADR @ Binary code to CONTinue at
1887            **         Set the SUSP annunciator (SFLAGs)
1888            **
1889            **
1890            ** What registers/RAM may be used if not handled?:
1891            **      --A-C, D[15-5] D0, D1, P always available (FPOLL only)--
1892            **      --NOTE: D[A] is sacred in FPOLL!!--
1893            **      --R1-R4, ST (low 12), scratch RAM--
1894            **      This is a top level poll ... nothing else is going on
1895            **
1896            ** Envisioned application(s):
1897            **      Implement BREAKPOINT capability within program:
1898            **        Set sExcept at entering to allow checking after
1899            **        each statement
1900            **
1901            **      Indicator to FORTH/VISICALC type applications that
1902            **      BASIC has been invoked.
1903            **
1904            ** History:
1905            **
1906            **    Date     Programmer             Modification
1907            **  --------   ----------   ----------------------------------
1908            **  01/16/83   JP           Added Poll
1909            **  04/23/83   JP           Updated/revised documentation
1910            **
1911            ********************************************************************
1912            ********************************************************************
1913            ********************************************************************
1914            ********************************************************************
1915            **
1916            ** Name:(S) pBSCex  -  Poll to Exit BASIC Interpreter
1917            **
1918            ** Category:   POLL
1919            **
1920            ** Type:       FPOLL
1921            **
1922            ** Purpose:
1923            **      Fast Poll when Exiting the BASIC interpreter
1924            **      Indicates program/statement execution is stopping
1925            **      Caused from:
1926            **        End of line of statement execution
1927            **        Program ENDing or STOPping
1928            **        Halt from:  ATTN key, SST, PAUSE, Error
1929            **        Ending a Binary program
1930            **      System is about to return to MAINLP
1931            **
1932            **
1933            ** Should poll be "Handled" (return with XM=0)?:
```

```
1934            **      No - This poll should never be "handled"
1935            **      Either the LEX file "TAKES OVER"
1936            **      or responds "not handled" so other LEX files may
1937            **      respond.
1938            **
1939            ** Meaning of "Handling" Poll (what does code do if handled?):
1940            **      Clear/update information
1941            **      If TAKE-OVER ---> gain control after BASIC execute
1942            **                        before we go back to MAIN LOOP
1943            **
1944            ** Entry conditions for handler (registers, ST, RAM, etc.):
1945            **      --Carry set on entry iff fastpoll--
1946            **      B[A] = Poll number (pBSCex)
1947            **      R2(A)= Filetype
1948            **      HEX mode.
1949            **      P=0.
1950            **      Math stacks have been collapsed
1951            **      Exceptions are checked PRIOR to this poll
1952            **        See pExcpt Poll
1953            **      If not Error Exit (not sERROR)
1954            **        Buffers have been flushed
1955            **      If NoCont (S14):
1956            **         If Program was running (and BASIC file)
1957            **            SUSP is lit
1958            **            CNTADR updated
1959            **            CURRL updated
1960            **         Halting due to one of the following:
1961            **            ATTN Key   (ATNFLG RAM is non-zero)
1962            **            END/STOP or end of program  (sENDx=1)  (S1)
1963            **            Error      (sERROR=1) (S0)
1964            **            SST
1965            **            PAUSE
1966            **            END(DEF), END(SUB), RETURN from keyboard
1967            **          (Error Exits can be trapped with pERROR,pWARN polls)
1968            **      If not NoCont (S14=0)
1969            **         If PgmRun (S13) --> Program was running
1970            **         sENDx=1 if STOP/END statement
1971            **
1972            **
1973            ** NOTE:
1974            **
1975            **      GOSUB, CALL, FNx invoked from the keyboard
1976            **      will enter and exit as Keyboard Execute...
1977            **      The PgmRun flag will NOT be set!!!!
1978            **      RETURN,ENDSUB,ENDDEF clear PgmRun before exiting
1979            **
1980            **      For CALL: pCALRS polls when CALL is ending
1981            **      For FNx:  pFNOUT polls when FNx  is ending
1982            **
1983            ** Normal exit conditions from handler if not handled (ST, RAM,
1984            ** registers, etc.):
1985            **      HEX mode.
1986            **      XM=1.
1987            **      Preserve sENDx, sERROR, PgmRun, NoCont, ATNFLG
1988            **
```

```
1989            ** Available subroutine levels: 6
1990            **      --FPOLL handler is two levels deeper than caller--
1991            **      This is a top level Poll --- 6 levels available, unless
1992            **      TAKING OVER... then 7
1993            **
1994            ** What registers/RAM may be used if not handled?:
1995            **      --A-C, D[15-5] D0, D1, P always available (FPOLL only)--
1996            **      --NOTE: D[A] is sacred in FPOLL!!--
1997            **      --R0-R4, scratch RAM--
1998            **
1999            ** Special Considerations:
2000            **    Binary Files may return through this exit point
2001            **
2002            **    A Binary file taking an error exit through the mainframe
2003            **    can "SUSPend" a binary program by setting CNTADR at
2004            **    the address to continue at within the file and setting
2005            **    the SUSP annunciator.
2006            **
2007            ** Envisioned application(s):
2008            **      Allow a LEX file to gain control after BASIC execution
2009            **
2010            ** History:
2011            **
2012            **    Date      Programmer              Modification
2013            **  --------    ----------   ----------------------------------
2014            **  07/20/82    JP           Added poll/documentation
2015            **  01/16/82    JP           Modified poll
2016            **  04/23/82    JP           Revised/updated documentation
2017            **  04/25/82    JP           Pass filetype in R2
2018            **
2019            ************************************************************
2020            ************************************************************
2021            ************************************************************
2022            ************************************************************
2023            **
2024            ** Name:(S) pExcpt  -  Poll on Exception after Stmt Execute
2025            **
2026            ** Category:   POLL
2027            **
2028            ** Type:       FPOLL
2029            **
2030            ** Purpose:
2031            **      Fast poll to indicate Exception has occured
2032            **      Allows servicing of Exceptions at the end of each
2033            **        statement execute.
2034            **      The Exception flag (Except (S12)) must have been set
2035            **        in response to pSREQ or prior to re-entry to BASIC
2036            **        loop (@ RUNRT1)
2037            **
2038            ** Should poll be "Handled" (return with XM=0)?:
2039            **      NO - This poll must continue to all LEX files
2040            **
2041            ** Meaning of "Handling" Poll (what does code do if handled?):
2042            **      You can process YOUR exception, but indicate the Poll
2043            **      was NOT handled.
```

```
2044            **
2045            ** Entry conditions for handler (registers, ST, RAM, etc.):
2046            **      --Carry set on entry iff fastpoll--
2047            **      B[A] = Poll number (pExcpt)
2048            **      HEX mode.
2049            **      P=0.
2050            **      Except (S12) = 0 from Mainframe
2051            **        Subsequent "responders" may set this to cause
2052            **        Except next time around.
2053            **      PgmRun = 1
2054            **        If program running
2055            **      NoCont = 1 (S14)
2056            **        If execution NOT to continue
2057            **           Caused from SST, PAUSE, END/STOP,
2058            **           END(DEF), END(SUB), RETURN from Keyboard
2059            **      The attention key HAS NOT been checked, yet
2060            **        ATNFLG RAM location#0 if ATTN Key hit
2061            **        The ATTN Key will cause program execution to stop
2062            **      DSPSTA (RAM) holds S0-S11
2063            **        sENDx  = 1 (S0) if END/STOP or End of Program
2064            **      RSTK(3)  Third Return Stack Level (0,1,2)
2065            **             = D0 setting from RUNRTN
2066            **               Points at EOL or @ following statement just
2067            **               executed.
2068            **
2069            ** Normal exit conditions from handler if handled (ST, RAM,
2070            ** registers, etc.):
2071            **      Response to this poll should NEVER indicate "handled
2072            **
2073            ** Normal exit conditions from handler if not handled (ST, RAM,
2074            ** registers, etc.):
2075            **      HEX mode.
2076            **      XM=1.
2077            **      S12-S15 must be preserved
2078            **      Stack levels: 0,1,2  preserved
2079            **
2080            ** Available subroutine levels: 4
2081            **      --FPOLL handler is two levels deeper than caller--
2082            **      This poll is issued from the TOP level
2083            **      But the Current D0 is 3rd level on stack
2084            **      This value cannot be lost; nor the return address to
2085            **      FPOLL
2086            **      Preserve levels: 0,1,2
2087            **
2088            ** NOTE:
2089            **      Error Exit to BASIC loop does NOT check exceptions
2090            **      Low status are restored from DSPSTA at the End of the
2091            **        Poll.
2092            **      Math stack has been collapsed
2093            **      ATTN key has been checked---causing NoCont to set (S14)
2094            **      Timers (1-3) will be checked after the pExcept poll
2095            **
2096            ** What registers/RAM may be used if handled?:
2097            **      N/A
2098            **
```

```
2099                ** What registers/RAM may be used if not handled?:
2100                **      --A-C, D[15-5] D0, D1, P always available (FPOLL only)--
2101                **      --NOTE: D[A] is sacred in FPOLL!!--
2102                **      --R0-R4, S0-S11
2103                **
2104                ** Envisioned application(s):
2105                **      Service external alarms/timers
2106                **      Service ON INTR statement
2107                **      Implement BREAKPOINT capability in BASIC
2108                **        Checking next statement to execute for Breakpoint
2109                **        Setting Except (S12) so pExcpt will occur at the end
2110                **        of the next statement
2111                **      Servicing ON TIMER# > 3 from an extended statement
2112                **
2113                ** History:
2114                **
2115                **    Date     Programmer            Modification
2116                **  --------   ----------   ----------------------------------
2117                **  01/16/83   JP           Added poll
2118                **  04/04/83   JP           Status saved/restored in DSPSTA
2119                **  05/07/83   JP           Updated documentation header
2120                **  05/18/83   JP           Attn Key not checked before poll
2121                **
2122                ************************************************************
2123                ************************************************************
2124                ****************************************
2125                ****************************************
2126                *
2127                * Main Entry into BASIC Interpreter
2128                *   BSCEXC:  Clear No Continue flag
2129                *   BSCEX2:  RUN/CONT/SST entry
2130                *
2131                *   PgmRun = 1 if beginning to execute a program
2132                *   If PgmRun
2133                *      D0 @ EOL before statement to execute
2134                *   else
2135                *      D0 @ Statement length byte
2136                *
2137                ****************************************
2138                ****************************************
2139                *
2140                * Move D0 into R0 for Fast Poll
2141                * Poll on entry into BASIC interpreter
2142                *
2143 07437 84E    =BSCEXC ST=0   NoCont          Clear NoCont flag
2144 0743A 136    =BSCEX2 CDOEX
2145 0743D 108            R0=C                  Move D0 to R0
2146 07440 7CD1           GOSUB  FPoll          Fast Poll
2147 07444 5F             CON(2) =pBSCen        Entering BASIC interpreter
2148 07446 118            C=R0
2149 07449 134            D0=C                  Restore D0
2150                *
2151                * If statement execute (not program running)
2152                *    No line number to skip
2153                *    go update PCADDR @ statement length byte
```

```
2154                *
2155 0744C 86D             ?ST=0  PgmRun          not Running ?
2156 0744F 43              GOYES  BSCX30
2157                *
2158                * else
2159                *    DO @ EOL or @ (multi-statement line)
2160                *       Check for EOL
2161                *       If @
2162                *          go update PCADDR
2163                *
2164                * LABEL entry if Label within multi-statment line
2165                *    Avoids display of intermediate labels during SST]
2166                *
2167 07451 14A   =BSCXLP A=DAT0 B              Look for EOL, @ or ELSE
2168 07454 161    BSCX10 DO=DO+ 2              Move past token
2169 07457 20              P=     0
2170 07459 90C             ?A#0   P            Not EOL ?
2171 0745C 72              GOYES  BSCX30
2172                *
2173                * At EOL
2174                *    If from keyboard   (not running (PgmRun))
2175                *       go Read filetype and exit BASIC
2176                *    else
2177                *       If @ end of current program scope
2178                *          go execute END statement
2179                *
2180 0745E 87D    BSCX20 ?ST=1  PgmRun          Running ?
2181 07461 60              GOYES  BSCX22        Check if @ end of program
2182 07463 6B61            GOTO   BSCEXT        Read filetype,Poll,goto MAINLP
2183 07467 1F76   BSCX22 D1=(5) =PRGMEN        End of current program
           5F2
2184 0746E 147             C=DAT1 A
2185 07471 132             ADOEX                Current position in program
2186 07474 8B2             ?A<C   A             Within program scope ?
2187 07477 60              GOYES  BSCX25        Yes
2188 07479 61D1            GOTO   END10         No - Execute END statement
2189 0747D 132    BSCX25 ADOEX                  Restore DO
2190 07480 163             DO=DO+ 4             Skip over line#
2191                *
2192                *   DO = Start of statement line  (@ line length byte)
2193                *   Update current PC address @ statement length byte
2194                *
2195 07483 1F97   BSCX30 D1=(5) =PCADDR
           6F2
2196 0748A 136             CDOEX
2197 0748D 145             DAT1=C A
2198 07490 134             DO=C                 SAVE PTR TO L.L. BYTE
2199 07493 161             DO=DO+ 2             Move past next token
2200 07496 08              CLRST                Clear status before jump
2201 07498 D0              A=0    A
2202 0749A 14A             A=DAT0 B             Read Begin BASIC token
2203 0749D 20              P=     0
2204                *
2205                * Check for Implied LET | FN
2206                *
```

```
2207 0749F 315B  BSCX40 LC(2)  =BASICs        Begin BASIC start
2208 074A3 9E2          ?A<C   B              Not Begin BASIC token ?
2209 074A6 91           GOYES  BSCX45         Implied LET Execute
2210             *
2211             * Calculate Execution Address
2212             *
2213 074A8 161          D0=D0+ 2              Move past BASIC token
2214 074AB 8E00         GOSUBL =EXCADR        Get Execution Address
           00
2215 074B1 06           RSTK=C
2216 074B3 01           RTN                   Jump to execution routine
2217             *
2218             * All statements return through NXTSTM or RUNRTN
2219             *
2220             *
2221             * LET Statement Execute
2222             * Implied LET execute
2223             *   Skip to next statement
2224             *   goto NXTSTM instead of LNSKP- to clear sEND flag
2225             *     will return to RUNRTN
2226             *
2227 074B5 0000         REL(5) =LETDC         LET statement Decompile
           0
2228 074BA 0000         REL(5) =LETP          LET statement Parse
           0
2229 074BF      =LET
2230 074BF 8F00  BSCX45 GOSBVL =ASNMNT
           000
2231 074C6 6E02         GOTO   Nxtstm         Skip over remarks
2232             *
2233             *
2234             * Jump back from Exception Checking
2235             *
2236             * CKEX-0:  (No exceptions or service requests)
2237             *
2238             *   If pending timer
2239             *      Save D0 on stack
2240             *      go service timer
2241             *   else
2242             *      go Continue BASIC loop
2243             *
2244             * CKEX00:  (Error Exit)
2245             *
2246             *   go Continue BASIC loop
2247             *
2248 074CA 1E16  CKEX-0 D1=(4) =PNDALM        Pending alarm bits
           7F
2249 074D0 14F          C=DAT1 B
2250 074D3 A66          C=C+C  B              Shift off Bit 4 (Timeout bit)
2251 074D6 90E          ?C#0   P              Timer 1 or 2 or 3 set ?
2252 074D9 60           GOYES  CKEX01         Save D0 & go Service timers
2253 074DB 6AE0  CKEX00 GOTO   BSCX60         Nothing to service
2254 074DF 136   CKEX01 CDOEX
2255 074E2 06           RSTK=C                Save D0 on Stack
2256 074E4 425          GOC    CKEX08         B.E.T. - go service timers
```

```
2257                *
2258                * Most statements return to RUNRTN through GOLONG NXTSTM
2259                *   sENDx is clear from NXTSTM (unless END statement)
2260                *   sENDx set if END Statement; NoCont set if END within Program
2261                *   DO @ EOL | @ of next statement
2262                *
2263                * GOTO, GOSUB, CALL, END SUB, FN return to RUNRT1
2264                *   DO @ EOL | @ of next statement
2265                *
2266                * Error Exits return from MFERR/BSERR and TRANSFORM to ERRRTN
2267                *   DO @ EOL | @ before current statement in error
2268                *   sENDx is clear
2269                *   NoCont is set
2270                *   sERROR is set
2271                *
2272                * NoCont = 1 if Don't Continue Execution (S14)
2273                *
2274                * Collaspe Math stack
2275                *
2276 074E7 841     =RUNRT1 ST=0   sENDx          Clear END execute flag
2277 074EA 840     =RUNRTN ST=0   sERROR         Clear ERROR exit flag
2278 074ED 20      =ERRRTN P=     0
2279 074EF 1FE9           D1=(5) =FORSTK
           5F2
2280 074F6 147            C=DAT1 A              FORSTK pointer
2281 074F9 1C4            D1=D1- 5              MTHSTK/AVMEME
2282 074FC 145            DAT1=C A              Collaspe MTHSTK --> FORSTK
2283                *
2284                * If Error Exit
2285                *    Skip exception checking
2286                *    Avoids servicing an ON TIMER or ON INTR after an Error
2287                *      Tracing shows branching after Error occurs
2288                *    MAINLP will check Exceptions
2289                *
2290 074FF 870            ?ST=1  sERROR         Not continuing execution ?
2291 07502 9D             GOYES  CKEX00         Skip exception checking
2292                *
2293                * If no exceptions and no Hardware Service Request
2294                *    go Check if Pending Timers
2295                *
2296 07504 87C            ?ST=1  Except         Exception occured ?
2297 07507 D0             GOYES  CKEX05
2298 07509 80E            SREQ?                 Service request ?
2299 0750C 834            ?SR=0                 None ?
2300 0750F BB             GOYES  CKEX-0         Continue
2301 07511 824            SR=0                  Leave Service request clear
2302 07514 136   CKEX05 CDOEX
2303 07517 06             RSTK=C                Save DO on stack
2304 07519 8E00           GOSUBL =CKSREQ        Check service requests
           00
2305 0751F 87C            ?ST=1  Except         Exceptions ?
2306 07522 60             GOYES  CKEX07
2307 07524 6990  CKEX06 GOTO   BSCX50          Restore DO and continue
2308                *
2309                * Exception occured
```

```
2310            *   Fast Poll for Exception
2311            *   Restore S0-S11 to preserve sENDx setting
2312            *
2313 07528 84C   CKEX07 ST=0   Except         Clear Exception flag
2314 0752B 71F0         GOSUB  FPoll          Fast poll
2315 0752F 8F           CON(2) =pExcpt        Exception occured
2316 07531 8E00         GOSUBL =USRSTA        Restore status from DSPSTA
          00
2317            *
2318            * If Attention Key hit
2319            *    Avoid Timer checks
2320            *
2321            * If program running
2322            *    Set mask for checking timers
2323            *        0 = Allows for shift before test
2324            *        E = 1110   Bit 0 = Timer 1
2325            *        D = 1101   Bit 1 = Timer 2
2326            *        B = 1011   Bit 2 = Timer 3
2327            *        0 =        Done
2328            *    Read Pending alarm bits
2329            *    Count Timer# and check if bit set for timer
2330            *
2331 07537 7271  CKEX08 GOSUB  CK"ON"         Set NoCont if ATTN hit.
2332 0753B 58E          GONC   CKEX06         If ATTN Key/ skip timer check
2333 0753E 86D          ?ST=0  PgmRun         Not running a program ?
2334 07541 D7           GOYES  BSCX50         Continue
2335 07543 340E         LCHEX  0BDE0          Mask to check Timer bits
          DB0
2336 0754A D5           B=C    A              Move mask to B
2337 0754C AC0          A=0    S              Clear Timer#
2338 0754F 1F16  CKEX10 D1=(5) =PNDALM        Pending Alarm RAM
          7F2
2339 07556 14F          C=DAT1 B              Read pending alarm bits
2340 07559 F5    CKEX20 BSR    A              Shift to next mask
2341 0755B 909          ?B=0   P              All 3 timers checked ?
2342 0755E 06           GOYES  BSCX50         Done, continue
2343 07560 B44          A=A+1  S              Checking next timer #
2344 07563 0E01         B=B&C  P              Turn off timer bit
2345 07567 901          ?B=C   P              Was Timer never on ?
2346 0756A FE           GOYES  CKEX20         Check next timer
2347            *
2348            * Get Corresponding ON TIMER address for Timer#
2349            * If non-zero TIMER address
2350            *    Check if TIMER address within current Program Scope
2351            *    If within scope
2352            *       Update PNDALM to zero Timer bit
2353            *       Set Except if another timer is due
2354            *         Enables timer to be detected next statement
2355            *       C <-- ON TIMER address
2356            *       Set ONTIMER statement flag
2357            *       Clear ONERR flag
2358            *       RSTK = Next statement to execute after Timer
2359            *       A(S) = Timer #
2360            *       go process ON TIMER statement
2361            * else
```

```
2362                *    go check if any other Timers went off
2363                *
2364 0756C 8E00            GOSUBL =GTMRA+        Get ON TIMER address
          00
2365 07572 143             A=DAT1 A              Read TIMER address
2366 07575 8A8             ?A=0   A              No TIMER address ?
2367 07578 7D              GOYES  CKEX10         go Check if any other timers
2368 0757A 7441            GOSUB  Scopck         Check program scope
2369 0757E 40D             GOC    CKEX10         Not within current program
2370 07581 100             R0=A                  Save Timer#, ON TIMER address
2371 07584 D4              A=B    A              Retrieve Timer bit mask
2372 07586 1F16            D1=(5) =PNDALM
          7F2
2373 0758D 1590            DAT1=A 1              Update Alarm RAM; zero Timer bit
2374 07591 8F00            GOSBVL =ALMSRV        Set Except if another timer due
          000
2375                *
2376                * If tracing
2377                *    Update PCADDR at statement about to execute
2378                *    Allows TRACE with Timers to look right
2379                *
2380 07598 75B5            GOSUB  TrfCk-         Tracing ?
2381 0759C 401             GOC    CKEX30         No
2382 0759F 07              C=RSTK                Next stmt address
2383 075A1 06              RSTK=C
2384 075A3 134             D0=C                  D0 @ EOL or @
2385 075A6 163             D0=D0+ 4              Move into statement
2386 075A9 7BA5            GOSUB  UPDPC          Update PCADDR @ start of LINE
2387                *
2388                *  Set ONTIMER statement flag
2389                *  Clear ONERR flag
2390                *  RSTK = Next statement to execute after Timer
2391                *  A(S) = Timer #
2392                *  go process ON TIMER statement
2393                *
2394 075AD 110     CKEX30 A=R0                  Restore Timer#, ON TIMER address
2395 075B0 D6              C=A    A              Move ON TIMER address to C
2396 075B2 856             ST=1   sONTMR         Set ON TIMER statement flag
2397 075B5 844             ST=0   sONERR         Clear ON ERROR statement flag
2398 075B8 8C00            GOLONG =ONTIMR        Process ON TIMER statement
          00
2399                *
2400                * Exceptions have been checked
2401                * Restore D0
2402                * Clear sERROR flag incase set by Exception Checking
2403                *
2404 075BE 07      BSCX50 C=RSTK                Restore D0
2405 075C0 134             D0=C
2406 075C3 840             ST=0   sERROR         Clear ERROR flag
2407                *
2408                * If continue execution
2409                *    go process end of statement & possible next line
2410                *
2411 075C6 87E     BSCX60 ?ST=1   NoCont         Don't continue ?
2412 075C9 60              GOYES  BSCX70
```

```
 2413 075CB 658E          GOTO   BSCXLP
 2414              *
 2415              * Don't Continue execution
 2416              *
 2417              *
 2418              * BSCEXT:  Exit BASIC Interpreter
 2419              *
 2420              *   sERROR (S0) = 1 if Error occured
 2421              *
 2422              *   Return to Keyboard Entry:  PgmRun=0
 2423              *     ENDSUB, END DEF and RETURN enter here if Keyboard Return
 2424              *
 2425              *   Binary file return point from ENDBIN
 2426              *   ENDALL return point:
 2427              *     PURGE/MERGE current running file
 2428              *
 2429              *   If PgmRun is set and BASIC program
 2430              *      D0 must be valid; since SUSP will occur
 2431              *      D0 @ statement to SUSPended
 2432              *
 2433              *
 2434              *   Clear Program Annunciator
 2435              *   Read filetype  (into R2)
 2436              *   If non BASIC file
 2437              *      Don't suspend --- go Exit BASIC loop
 2438              *   If not running
 2439              *      go exit BASIC
 2440              *   else
 2441              *      If not END statement
 2442              *         Update Continue address
 2443              *         Set SUSP Annunciator/Flag
 2444              *      Compute and Update Current Line#
 2445              *
 2446 075CF       =BSCEXT
 2447 075CF 7B01   BSCX70 GOSUB  SflgCp          Clear Program Annunciator
 2448 075D3 7711          GOSUB  RDCHD+          Read current file type
 2449 075D7 7361          GOSUB  BASCHK          BASIC file ?
 2450 075DB 4B2           GOC    BSCEX+          No !  R2=filetype
 2451 075DE 86D           ?ST=0  PgmRun          Not Running ?
 2452 075E1 62            GOYES  BSCEX+
 2453              *
 2454              * If not END/STOP statement
 2455              *    Restore D0
 2456              *    If next statement = ELSE --> Skip to EOL
 2457              *    Update Continue Address <-- D0
 2458              *        D0 @ EOL or @ before next statement
 2459              *    Set SUSP Annunciator/Flag
 2460              *
 2461 075E3 871           ?ST=1  sENDx           END statement ?
 2462 075E6 12            GOYES  BSCEX+
 2463 075E8 163           D0=D0+ 4               Position past @ and line length if
 2464 075EB 14A           A=DAT0 B               Read statement token OR part of lin
 2465 075EE 3100          LC(2)  =tELSE
 2466 075F2 183           D0=D0- 4               Position back to @ or EOL
 2467 075F5 966           ?A#C   B               Not ELSE ?
```

```
2468 075F8 80          GOYES  BSCX85
2469 075FA 8E00        GOSUBL =EOLSCN        Scan to EOL
           00
2470               *
2471               * Update Continue address
2472               * Set SUSP annunciator
2473               * Compute and update CURRL at line containing CONTinue address
2474               *
2475 07600 136   BSCX85 CDOEX                 Move addr to C
2476 07603 718D        GOSUB  CNTCUR         Update CNTADR, Set SUSP, Update CUR
2477               *
2478               * EXIT BASIC Interpreter
2479               *
2480               * SST @ End of Program Entry
2481               *    Executes END statement
2482               *    Avoids updating CURRL, which was set from prior PAUSE/SST
2483               *      PCADDR has no meaning in SST execute
2484               *
2485               * If not an error
2486               *    Flush all open buffers
2487               *      Cannot "try" to flush buffers if Error:
2488               *      If HPIL loop is broken
2489               *         Error is generated;
2490               *         Attempting to flush buffers on error causes Infinite lo
2491               * Fast Poll for exiting BASIC Interpreter
2492               * Clear Don't Continue flag
2493               * golong MAINLP
2494               *
2495 07607 870   BSCEX+ ?ST=1  sERROR         Error occured ?
2496 0760A 90          GOYES  BSCX95
2497 0760C 8F00        GOSBVL =FLUSHA        Flush all buffers
           000
2498 07613 7900  BSCX95 GOSUB  FPoll          Fast Poll
2499 07617 6F          CON(2) =pBSCex        Exiting BASIC interpreter
2500 07619 84E         ST=0   NoCont         Clear Don't Continue flag
2501 0761C 673C        GOTO   MLPEXT         Return to Main Loop
2502 07620 8D00 =FPoll GOVLNG =FPOLL
           000
```

```
2503                    STITLE END/STOP Execute
2504            ********************************************************************
2505            ********************************************************************
2506            **
2507            ** Name:    END     -  END, END ALL, END SUB, END DEF Statements
2508            ** Name:(S) ENDALL  -  External Stmt entry to perform END ALL
2509            ** Name:(S) ENDBIN  -  End Binary Program or Subprogram
2510            ** Name:    END10   -  STOP Statement Execute
2511            ** Name:    END20   -  END SUB reentry
2512            ** Name:    STOP    -  STOP Statement Execute
2513            ** Name:    EXITRN  -  Clear status, return to BASIC loop
2514            **
2515            ** Category:   STEXEC
2516            **
2517            ** Purpose:
2518            **      These entry points deal with terminating execution
2519            **      of in the current environment due to an explicit
2520            **      command such as END or STOP, or a SST past the last
2521            **      statement in the program.  The running program may
2522            **      be BASIC or Binary.
2523            **
2524            **      END checks for ALL token
2525            **          checks for ENDSUB/ENDDEF
2526            **          Returns to BASIC loop allowing exceptions to be
2527            **          checked
2528            **
2529            **      S2 set will cause ending of execution so that:
2530            **         Exceptions not checked
2531            **         Program not suspended, CNTADR not updated
2532            **
2533            **      All entries but ENDALL collapse ONLY ONE level
2534            **      ENDALL collapses to one level
2535            **
2536            ** Entry:
2537            **    END:        DO past END token
2538            **                Checks for ALL token
2539            **    STOP:
2540            **    ENDBIN:
2541            **    END10:      (Checks if END SUB or END DEF)
2542            **                (BASIC Loop entry if @ program scope end)
2543            **                (SST @ Program End entry)
2544            **
2545            **                sSST (S2=1) if non-exception/nonprogram exit
2546            **                     (Clears PgmRun (S13=0), Clears S0-S11)
2547            **                     (Returns to BASIC loop without checking
2548            **                      exceptions)
2549            **                     (Prevents update of Cont Addr [CNTADR]
2550            **                      and SUSPend of program)
2551            **                Collapse stacks one level ONLY
2552            **
2553            **      END20:    END SUB reentry
2554            **                sSST assumed cleared  (S2=0)
2555            **                If S2=1 acts like END10 entry
2556            **
2557            **      ENDALL:   External Statement entry
```

```
2558            **              Sets sSST (S2=1) to avoid CNTADR update and
2559            **                   program suspension
2560            **              Avoids checking of exceptions in BASIC loop
2561            **              Clean-up for TRANSFORM current file
2562            **              Clean-up for PURGE current file
2563            **              Collapse stacks down to ONE level
2564            **
2565            **      All entries, but ENDALL, collapse ONLY ONE level
2566            **
2567            ** Exit:
2568            **      If END ---> sENDx (S1=1) for BASIC loop return
2569            **                    Prevents SUSPend of program
2570            **                  Through NXTST1 to avoid sENDx clearing
2571            **                    Returns to BASIC loop so exceptions
2572            **                      are checked
2573            **                  NoCont (S14=1) if within program
2574            **                    Causes BASIC loop execution to stop
2575            **      If END DEF or implied END DEF
2576            **             ---> Through ENDDEF
2577            **      If END SUB or implied END SUB
2578            **             ---> Through ENDSB-
2579            **      If SST @ PRGMEN or non exception check END desired
2580            **             ---> Through BSCEXT with PgmRun (S13) clear
2581            **                  Exceptions are not checked
2582            **                  Prevents CNTADR update and prgm SUSPension
2583            **
2584            **      If non BASIC program
2585            **             ---> Through EXITRN
2586            **                  Clears S0-S11; exit BASIC loop (BSCEXT)
2587            **                  Exceptions are not checked
2588            **                  CNTADR not updated, program not SUSPended
2589            **
2590            ** Calls:      CLRSTK, CLOSEA, CLPSTK, GETSTC, SUBCHK
2591            **
2592            ** Uses:     A-D,P, D1, D0, CNTADR, sENDx (S1), sSST (S2),
2593            **           R0,R2,ALRM (+36), PNDALM (+1),STMTD1,f1SUSP,PgmRun
2594            **
2595            ** Stk lvls:   6
2596            **
2597            ** Algorithm:
2598            **        If END ALL
2599            **           goto ENDAL1:
2600            ** ENDBIN:
2601            ** END10: If END DEF | END SUB
2602            **           go process appropriate statement
2603            ** END20: Clear addresses, one level of stacks    (CLRSTK)
2604            ** END30: Close all open files                   (CLOSEA)
2605            **        If non BASIC file                      (GETSTC)
2606            **           go Clear status and Exit BASIC loop (BSCEXT)
2607            **        If non programmatic END desired        (sSST)
2608            **           Clear PgmRun to prevent SUSPend
2609            **           go Clear status and Exit BASIC loop (BSCEXT)
2610            **        else
2611            **           If program running
2612            **              Set Don't Continue flag          (NoCont)
```

```
2613            **          Set END Execute flag                (sENDx)
2614            **          Golong to end of BASIC loop through
2615            **                 NXTST1 to avoid sENDx clearing
2616            **
2617            ** ENDALL:Set sSST flag
2618            ** ENDAL1:Collaspe stacks to one level          (CLPSTK)
2619            **       goto END30
2620            **
2621            ** Note:
2622            **
2623            ** The sENDx flag was originally used to distinquished END from
2624            ** all other statements/conditions that stop the BASIC loop exec.
2625            ** If a program had been running,
2626            ** this flag allowed CURRL to be updated to the END statement,
2627            ** but prevented the SUSP annunciator from lighting and the
2628            ** CONTINUE address from being updated.
2629            **
2630            ** This sSST flag was used to avoid any checking of a program
2631            ** running by returning to a different place in the BASIC loop,
2632            ** since CURRL could not be updated in situations like SST past
2633            ** the program end.
2634            **
2635            ** When the decision was made to update CURRL only when
2636            ** SUSPending the use of two flags is not that different.
2637            ** A "normal" END statement returns through NXTSTM to the BASIC
2638            ** loop. This causes exceptions to be checked before execution is
2639            ** stopped if a program was running.  If from the keyboard,
2640            ** execution continues.  If sSST is set (from SST past
2641            ** the end of the program or for TRANSFORMing the current file...
2642            ** then the BASIC loop is reentered below the exception checking.
2643            **
2644            ** In either case, neither CNTADR is updated, nor SUSP lit.
2645            **
2646            ** History:
2647            **
2648            ** Date        Programmer         Modification
2649            ** ----------  -----------        -------------------------
2650            ** 03/08/83    JP                 STOP === END SUB, END DEF
2651            ** 03/17/83    JP                 Packed D1=(5) CALSTK
2652            ** 04/25/83    JP                 CLRST thru EXITRN if sSST
2653            ** 05/09/83    JP                 Clear PgmRun before EXITRN
2654            ** 05/17/83    JP                 Check ENDSUB/DEF if SST
2655            **                                   at end of program (PRGMEN)
2656            ** 06/05/83    JP                 END10 is Binary program return
2657            ** 06/05/83    JP                 If nonBASIC prgm---> EXITRN
2658            ** 06/05/83    JP                 ENDBIN entry point added
2659            **
2660            ************************************************************************
2661            ************************************************************************
2662            *
2663 07627 8D00 =DEFEND GOVLNG =ENDDEF
          000
2664            *
2665 0762E 8D00  SUBEND GOVLNG =ENDSB-
          000
```

```
2666                *
2667 07635 0000             REL(5) =ENDDC
           0
2668 0763A 0000             REL(5) =ENDP
           0
2669                *
2670                * Entry points:
2671                *   END     END statement
2672                *   ENDBIN
2673                *   END10   SST @ PRGMEN  (at end of program scope)
2674                *           Binary Program reentry point
2675                *           Allows  CALL Binary and RUN Binary
2676                *
2677 0763F 14A  =END     A=DATO B              Read next token
2678 07642 3100          LC(2)  =tALL
2679 07646 962           ?A=C   B              END ALL ?
2680 07649 45            GOYES  ENDAL1
2681                *
2682                * Binary Program Reentry point:
2683                *
2684                * If SST @ PRGMEN
2685                *    sSST = 1 to cause non-programmatic END
2686                *
2687 0764B        =ENDBIN
2688 0764B 1F7B =END10  D1=(5) =PRMPTR        SEE IF WE ARE IN A DEF FN
           5F2
2689 07652 14F           C=DAT1 B              READ PARAMETER COUNT
2690 07655 96E           ?C#0   B
2691 07658 FC            GOYES  DEFEND         IF PARAM COUNT #0, GOTO DEFEND
2692 0765A 8F00          GOSBVL =SUBCHK        SEE IF WE ARE STILL IN A SUB-PROG
           000
2693 07661 4CC           GOC    SUBEND
2694                *
2695                * Entry points:
2696                *   END20:   END program
2697                *            Rentry from ENDSUB  (sSST=0 if END)
2698                *                                (sSST=1 if SST @ PRGMEN)
2699                *            If not ENDSUB...perform END
2700                *   END30:   END ALL - CLPSTK done instead of CLRSTK
2701                *
2702                * Collapse FOR/NEXT Stack, GOSUB Stack
2703                * Clear CNTADR --> TMADR2
2704                * Clear SUSP Annunciator/Flag
2705                * Close all open files
2706                * If SST @ End of Program or non-exception check END desired
2707                *    Jump to RUNSST to avoid exception check, CNTADR update
2708                *    and SUSPend of program
2709                * else
2710                *    If program running
2711                *       Set Don't Continue flag
2712                *    Set END Execute Flag
2713                *    Jump to NXTST1          (avoid sENDx clearing)
2714                *
2715 07664        =END20
2716 07664 73D6  END25  GOSUB  =CLRSTK        Collapse one level ofstacks
```

```
2717 07668 8F00  END30  GOSBVL =CLOSER        Close all open files
           000
2718 0766F 73B0         GOSUB  GETSTC         Check file type
2719 07673 433          GOC    EXITRN         If nonBASIC, exit BASIC loop
2720 07676 872          ?ST=1  sSST           SST @ End of Prog or non-prgm END
2721 07679 B2           GOYES  RUNSST         Clr status, Exit BASIC loop
2722 0767B 86D          ?ST=0  PgmRun         Not running ?
2723 0767E 50           GOYES  END40
2724 07680 85E          ST=1   NoCont         Set Don't Continue flag
2725 07683 851   END40  ST=1   sENDx          Set END Execute flag
2726 07686 8C00         GOLONG =NXTST1        Next statement/avoid sENDx clear
           00
2727               *
2728               * STOP Statement Entry
2729               *
2730 0768C 0000         REL(5) =STOPDC        STOP Decompile
           0
2731 07691 0000         REL(5) =RTNCC         Parse
           0
2732 07696 64BF =STOP   GOTO   END10          Process STOP statement
2733               *
2734               * END ALL External Statement Entry
2735               *   Set sSST to avoid CNTADR update/Pgrm SUSPend at BASIC loop en
2736               *       and to avoid exception checking
2737               *
2738 0769A 852  =ENDALL ST=1   sSST           Set single step flag /non prgm END
2739 0769D 7886  ENDAL1 GOSUB  CLPSTK         Collapse down to 1 level
2740 076A1 56C          GONC   END30          B.E.T.
2741               *
2742               * SST @ end of Program or END ALL
2743               * Clear Program Running flag --- so BSCEXT is correct
2744               *   and won't SUSPend program or update CNTADR
2745               * Clear status so not contrued as Error exit
2746               * Exit BASIC loop, avoiding exception checking
2747               *
2748 076A4 84D   RUNSST ST=0   PgmRun         Clear Program Running flag
2749 076A7 08   =EXITRN CLRST
2750 076A9 652F         GOTO   BSCEXT         Exit BASIC loop
```

```
                        STITLE CK"ON" - Check ON/ATTN Key
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) CK"ON"  -  Check ON / ATTN Key
                **
                ** Category:   EXCUTL
                **
                ** Purpose:
                **      Check if ON/ATTN key hit    (CK"ON" entry)
                **      This routines needs to be called after
                **           each statement execute
                **
                ** Entry:
                **
                ** Exit:
                **      Carry set
                **        ATTN key Not hit
                **      Carry clear
                **        ATTN Key hit
                **        NoCont (S14) set if ATTN key hit
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Exclusive: A(S),D1,NoCont(S14)
                **  Inclusive: A(S),D1,NoCont(S14)
                **
                **  S14 = ATTN key hit, NoCont flag
                **
                ** Stk lvls:   0
                **
                ********************************************************************
                ********************************************************************
076AD 1F24 =CK"ON" D1=(5) =ATNFLG
           4F2
076B4 1534         A=DAT1 S
076B8 948          ?A=0   S              ATTN key hasn't been pressed ?
076BB 00           RTNYES
076BD 85E          ST=1   NoCont         Set Don't Continue flag
076C0 01           RTN                   Carry clear
076C2 8C00 =Scopck GOLONG =SCOPCK
           00
```

```
2792                    STITLE PAUSE execute
2793                ********************************************************************
2794                ********************************************************************
2795                **
2796                ** Name:    PAUSE   -  Statement Execution
2797                **
2798                ** Category:   STEXEC
2799                **
2800                ** Purpose:
2801                **      Execute PAUSE statement
2802                **
2803                ** Entry:
2804                **      DO past PAUSE token
2805                **
2806                ** Exit:
2807                **      Through NXTSTM
2808                **
2809                ** Calls:      None
2810                **
2811                ** Uses.......
2812                **  Exclusive: S14 (NoCont)
2813                **  Inclusive: S14 (NoCont)
2814                **
2815                ** Stk lvls:   0
2816                **
2817                ** Detail:   Set Don't Continue flag (S14)
2818                **           Skip over PAUSE statement & return to RUN loop
2819                **
2820                **           RUN loop processing will cause a SUSP in the prog
2821                **           by S14 (NoCont) being set.
2822                **
2823                ********************************************************************
2824                ********************************************************************
2825 076C8 0000            REL(5) =STOPDC        Decompile
          0
2826 076CD 0000            REL(5) =RTNCC         Parse
          0
2827 076D2 85E   =PAUSE    ST=1   NoCont         Set Don't Continue flag
2828 076D5 8C00  =Nxtstm   GOLONG =NXTSTM        Skip STOP stmt and return to RUN lo
          00
2829                *
2830                *
2831 076DB 84D   =NOPRGM   ST=0   PgmRun         Clear Program Running flag
2832 076DE 312C  =SflgCp   LC(2)  =flPRGM        Clear program annunciator
2833 076E2 8C00  =SflagC   GOLONG =sflagc
          00
2834 076E8 8C00   Poll     GOLONG =POllj
          00
```

```
2835                    STITLE Read File Header
2836            ********************************************************************
2837            ********************************************************************
2838            **
2839            ** Name:(S) RDCHDR  -  Read Current File header, File length
2840            ** Name:(S) RDCHD+  -  Read Current File header,File length and t
2841            ** Name:(S) RDHDR1  -  Read File header, File length
2842            **
2843            ** Category:   FILUTL
2844            **
2845            ** Purpose:
2846            **      Read file header, return File length, possibly File type
2847            **
2848            ** Entry:
2849            **      RDCHDR:  Sets D1 = Start of Current File @ Header
2850            **               Assumes:
2851            **                  If P=0; File type read into R2
2852            **                  If P#0; File type not read into R2
2853            **      RDHDR1:  D1 @ Start of File @ header
2854            **               Assumes:
2855            **                  If P=0; File type read into R2
2856            **                  If P#0; File type not read
2857            **      RDCHD+:  Set D1 = Start of Current File
2858            **               Explicitly sets P=0
2859            **               File type will be returned in R2
2860            **
2861            ** Exit:
2862            **      Carry Clear
2863            **      D1 @ File length of header
2864            **       A = File length
2865            **           Current D1 + (A) = Next File in Chain
2866            **
2867            **      If P=0
2868            **         R2 = File type
2869            **
2870            **      P is NOT reset; necessary for GETSTC to call RDHDR1
2871            **      Calling routine must reset P=0 if desired.
2872            **
2873            ** Calls:      None
2874            **
2875            ** Uses.....
2876            **  Exclusive: A(A),P,R2 (if P=0),D1
2877            **  Inclusive: A(A),P,R2 (if P=0),D1
2878            **
2879            ** Stk lvls:   0
2880            **
2881            ** Detail:   File Header Format:
2882            **
2883            **           File Name      16 nibbles
2884            **           File Type       4
2885            **           Flags           2
2886            **           Creation Time   4
2887            **           Creation Date   5
2888            **           File Chain      5
2889            **           Implementation  8
```

```
2890                **
2891                **
2892                ** History:
2893                **
2894                **    Date      Programmer            Modification
2895                **  --------    ----------    ----------------------------------
2896                **  06/30/82    JP            Modified Documentation
2897                **  01/04/83    JP            Change S9 usage to P=0/P#0
2898                **
2899                ********************************************************************
2900                ********************************************************************
2901                *
2902 076EE 20     =RDCHD+ P=     0               Indicate File TYpe wanted
2903 076F0 1FD5 =RDCHDR D1=(5) =CURRST
           5F2
2904 076F7 143           A=DAT1 A               Start of CURRent file
2905 076FA 131    RDHDR0 D1=A
2906 076FD 17F   =RDHDR1 D1=D1+ oFTYPh          Skip to File Type
2907 07700 D0            A=0    A
2908 07702 880           ?P#    0               File type not wanted ?
2909 07705 90            GOYES  RDHD10
2910 07707 15B3          A=DAT1 4               Read File type
2911 0770B 102           R2=A                   R2 = File type
2912 0770E 17F    RDHD10 D1=D1+ (oFLENh)-(oFTYPh)  Move to File Length from File T
2913 07711 143           A=DAT1 A               Read File length
2914 07714 03            RTNCC
```

```
2915                    STITLE Get first line of BASIC file & EOF
2916            ********************************************************************
2917            ********************************************************************
2918            **
2919            ** Name:(S) GETSTC  -  Get Start/EOF Curr File/check Filetype
2920            ** Name:(S) GETST-  -  Get Start/EOF Curr File/don't check Filety
2921            ** Name:(S) GETST*  -  Get Start/EOF any file/check Filetype
2922            ** Name:    GETSTe  -  Get start/EOF Curr File/Error exit not BAS
2923            ** Name:    GETPeF  -  Check protection & get file start/EOF
2924            **
2925            ** Category:   FILUTL
2926            **
2927            ** Purpose:
2928            **      GETSTC,GETSTe:
2929            **        Return first line of BASIC/Binary file & EOF
2930            **        If P=0
2931            **           Verify that file is BASIC, Error Return if NOT
2932            **
2933            **        See GETSTe for Error Exit if non BASIC file
2934            **
2935            **      GETPeF:
2936            **        Check File protection
2937            **          Error exit if file protected
2938            **        Fall into GETSTC code
2939            **          Get start/end of BASIC file
2940            **          Error return if non BASIC file
2941            **
2942            ** Entry:
2943            **
2944            **      GETPeF: Checks file protections
2945            **              Falls into GETSTC
2946            **      GETSTC: D1 gets set to start of Current file
2947            **              Sets P=0
2948            **                   File type read into R2; Check if BASIC
2949            **                   Falls into BASCHK
2950            **      GETST-: D1 gets set to start of Current File
2951            **              Assumes P set on entry
2952            **                   Used for P#0 entry
2953            **                   File type not read into R2, not checked
2954            **      GETST*: A @ Start of file
2955            **              Assumes P value on entry
2956            **
2957            **      GETST1: D1 @ File length field of file
2958            **              A(A) contains file length
2959            **              If P=0
2960            **                 Checks file type in R2 for BASIC file type
2961            **
2962            ** Exit:
2963            **      If GETPeF entry:
2964            **         If file protected:
2965            **            Error Exit to MFERR   (eFPROT)
2966            **
2967            **       P=0
2968            **       D0  @ First line of file (at initial tEOL)
2969            **       D   = End of file
```

```
2970            **       A   = File length
2971            **
2972            **       If P#0
2973            **          Carry Clear
2974            **          File type NOT in R2, file type NOT checked
2975            **
2976            **       If P=0
2977            **          Fall into BASCHK
2978            **          If BASIC filetype
2979            **             Carry Clear
2980            **             R2  = File type
2981            **          else
2982            **             Error Return - C(0-4) = eFTYPE
2983            **
2984            ** Calls:      RDCHDR, RDHDR1, GETPRO (GETPeF entry only)
2985            **
2986            ** Uses.......
2987            **  Exclusive: A(A),C(A),D(A),D0,D1,P,R2 (if P=0)
2988            **  Ixclusive: A(A),C(A),D(A),D0,D1,P,R2 (if P=0)
2989            **
2990            ** Stk lvls:   GETSTC,GETPeF,GETST1,GETST*,GETST1: 1
2991            **             GETSTe:        2
2992            **
2993            ** Detail:
2994            **           Positions to first line of file assuming:
2995            **
2996            **           oBSsod = Offset to BASIC start of data, which
2997            **                    includes the permanent EOL.
2998            **
2999            **           Must subtract length of EOL to position @ first line
3000            **
3001            ** History:
3002            **
3003            **    Date     Programmer              Modification
3004            **  --------   ----------   ---------------------------------
3005            **  06/30/82   JP           Modified documentation
3006            **  09/15/82   JP           Changed to Error Return, not Exit
3007            **  01/04/83   JP           Changed S9 usage to P=0/P#0
3008            **  03/01/83   JP           Added GETPeF entry point
3009            **
3010            ************************************************************************
3011            ************************************************************************
3012            *
3013 07716 70EF =GETST* GOSUB  RDHDRO
3014 0771A 511          GONC   GETST1          (B.E.T.)
3015            *
3016 0771D 8E00 =GETPeF GOSUBL =GETPRO
           00
3017 07723 424          GOC    Mferr           Private file ?
3018            *
3019 07726 20   =GETSTC P=     0               Request file type in R2
3020 07728 74CF =GETST- GOSUB  RDCHDR          Read Current file header
3021 0772C 137  =GETST1 CD1EX                  C @ File length nib of file
3022 0772F 134          D0=C
3023 07732 C2           C=C+A  A
```

```
3024 07734 D7          D=C    A              D = End of file
3025 07736 16E         D0=D0+ (oBSsod)-1EOL  Offset to data - Perm EOL
3026 07739 880         ?P#    0              No file type to check ?
3027 0773C 41          GOYES  BASCK3
```

```
3028                    STITLE Verify BASIC file type
3029            ****************************************************************
3030            ****************************************************************
3031            **
3032            ** Name:(S) BASCHK  -  Verify File Type in R2 is BASIC
3033            ** Name:(S) BASCHA  -  Verify File Type in R2 is BASIC
3034            **
3035            ** Category:   FILUTL
3036            **
3037            ** Purpose:
3038            **      BASCHK:
3039            **        Verify that File type in R2(A) is BASIC
3040            **      BASCHA:
3041            **        Verify that File type in A(A) is BASIC
3042            **      Error return if not
3043            **
3044            ** Entry:
3045            **      P=0
3046            **      BASCHK:   R2(A) = File type
3047            **      BASCHA:   A(A)  = File type
3048            **
3049            ** Exit:
3050            **      P      =  0
3051            **
3052            **      If File type = BASIC
3053            **         Carry Clear
3054            **            R2(A) = File type
3055            **            A     = Preserved from Entry
3056            **      else
3057            **         Carry Set
3058            **            Error Return  C(0-4) = eFTYPE
3059            **            R2(A) = File type
3060            **            A(A)  = File type
3061            **
3062            ** Calls:      None
3063            **
3064            ** Uses.......
3065            **  Exclusive: C,R2
3066            **  Inclusive: C,R2
3067            **
3068            ** Stk lvls:   BASCHK: 0
3069            **             GETSTe: 2
3070            **
3071            ** Detail:     This code must IMMEDIATELY follow GETSTC
3072            **
3073            ** History:
3074            **
3075            **    Date     Programmer            Modification
3076            **  --------   ----------   ---------------------------------
3077            **  06/30/82   JP           Modified documentation
3078            **  09/15/82   JP           Changed to Error return/not exit
3079            **  12/17/82   JP           Added BASCHA entry
3080            **  01/04/83   JP           Added P=0 at end, due to GETSTC
3081            **  03/01/83   JP           Remove GETS-e entry due to NULLP
3082            **  04/25/83   JP           If non BASIC, R2 = filetype
```

```
3083                **
3084                **********************************************************************
3085                **********************************************************************
3086 0773E 122     =BASCHK AR2EX
3087 07741 3441    =BASCHA LC(5)  =fBASIC
           2E0
3088 07748 8A6             ?A#C   A
3089 0774B 90              GOYES  NOTBAS
3090 0774D 122             AR2EX
3091 07750 20       BASCK3 P=     0
3092 07752 03              RTNCC
3093 07754 3300     NOTBAS LC(4)  =eFTYPE        Illegal File type
           00
3094 0775A 102             R2=A                 R2 <-- Filetype
3095 0775D 02              RTNSC                Return with Carry set
3096                *
3097                * GETSTe - Calls GETSTC and Error Exits if NOT BASIC
3098                *
3099                *   Removed GETS-e entry since NULLP no longer needs it
3100                *
3101 0775F 73CF    =GETSTe GOSUB  GETSTC        Get current file start/end
3102 07763 500      GETSe1 RTNNC                File type is BASIC
3103 07766 8C00     Mferr  GOLONG =MFERR        Error Exit if not BASIC file
           00
```

```
3104                    STITLE GETPRe - Check File Type/Protection
3105            ****************************************************************
3106            ****************************************************************
3107            **
3108            ** Name:    GETPRe  -  Check Protection/Filetype & Error
3109            ** Name:    CHKPSF  -  Check if priv. or secure or non-BASIC file
3110            **
3111            ** Category:   FILUTL
3112            **
3113            ** Purpose:
3114            **      GETPRe:
3115            **        Read start and end of current file, checking file type
3116            **          Error exit if non-BASIC file
3117            **        Check protection on file
3118            **          Error exit if Private
3119            **
3120            **      CHKPSF:
3121            **        Error if non BASIC
3122            **        Error if PRIVATE
3123            **        Error if SECURE
3124            **
3125            ** Entry:
3126            **
3127            ** Exit:
3128            **      P       =  0
3129            **      GETPRe:
3130            **        Carry clear
3131            **          Current file is BASIC and NOT PRIVATE
3132            **          SB=1 if Write Protected file
3133            **          C(A) = eFPROT
3134            **        Error Exit      Through MFERR
3135            **          "eFTYPE" if non-BASIC file
3136            **          "eFPROT" if PRIVATE file
3137            **
3138            **      CHKPSF
3139            **        Carry set
3140            **          File is BASIC, not PRIVATE, UNSECURE
3141            **        Error Exit:
3142            **          "eFTYPE" if non BASIC
3143            **          "eFPROT" if PRIVATE or SECURE
3144            **
3145            **
3146            ** Calls:       GETSTe, GETPRO
3147            **
3148            ** Uses.......
3149            **  Exclusive: A(A),C(A),D(A),D0,D1,R2
3150            **  Inclusive: A(A),C(A),D(A),D0,D1,R2
3151            **
3152            ** Stk lvls:
3153            **             GETPRe: 3
3154            **             CHKPSF: 4
3155            **
3156            ** Detail:
3157            **
3158            ** Algorithm:
```

```
3159                ** 
3160                ** History:
3161                ** 
3162                **    Date     Programmer             Modification
3163                ** --------   ----------   ----------------------------------
3164                ** 10/25/82   JP           Added routine
3165                ** 03/02/83   JP           Added CHKPSF
3166                ** 
3167                ********************************************************************
3168                ********************************************************************
3169 0776C 7FEF =GETPRe GOSUB  GETSTe          Error if non BASIC file
3170 07770 8E00         GOSUBL =GETPRO         Get protection
           00
3171 07776 6CEF         GOTO   GETSe1          Error if non BASIC
3172                *
3173                * Check File type, Privacy, Security
3174                *
3175 0777A 7EEF =CHKPSF GOSUB  GETPRe          Error if non BASIC or PRIVATE
3176 0777E 832          ?SB=0                  not SECURE ?
3177 07781 00           RTNYES
3178 07783 52E          GONC   Mferr           Error exit if SECURE
```

```
3179                    STITLE Find Label in File
3180            ********************************************************************
3181            ********************************************************************
3182            **
3183            ** Name:(S) FINDLB  -  Find Label in Current Program
3184            ** Name:    ATCHK   -  Find Label in Current Program
3185            **
3186            ** Category:   EXCUTL
3187            **
3188            ** Purpose:
3189            **      Find Label in current program.  This routine is for run
3190            **      time only.  To find a label across a file call FCHLBL.
3191            **
3192            **      ATCHK: Late entry point to check if at an "@"
3193            **
3194            ** Entry:
3195            **      FINDLB:
3196            **        P=0
3197            **        B = Label to find
3198            **            Right justified with trailing blanks
3199            **              ("ABC" = 2020202020434241)
3200            **        File already chained
3201            **
3202            **      ATCHK:
3203            **        DO @ Possible "@" (multi-statement line)
3204            **
3205            ** Exit:
3206            **      FINDLB:
3207            **        P=0
3208            **        B = Label to find
3209            **        Carry Clear - Label found
3210            **          DO @ EOL or @ preceding the statement with Label
3211            **        Carry Set   - Label not found
3212            **      ATCHK:
3213            **        DO @ "@" or EOL
3214            **
3215            ** Calls:      LBLNAM
3216            **
3217            ** Uses.......
3218            **  Exclusive: A,C(A),DO
3219            **  Inclusive: A,C,DO,P
3220            **
3221            ** Stk lvls:   2
3222            **
3223            ** Detail:   Starting from label chain head (PRGMEN-5)
3224            **               Jump by Label Link looking for LABEL token
3225            **               When a LABEL token is found
3226            **                   Call LBLNAM to get label into A
3227            **                   If label matches the label in B
3228            **           ATCHK:     Position to EOL | @
3229            **                      Return CC
3230            **                   else
3231            **                      Continue until End of Label Chain reached
3232            **
3233            ** History:
```

```
3234                **
3235                **    Date     Programmer              Modification
3236                **  --------   ----------   ---------------------------------
3237                **  06/30/82   JP           Modified documentation
3238                **  04/08/83   JP           Test for @/line# using A(XS) = F
3239                **
3240                ********************************************************************
3241                ********************************************************************
3242                *
3243 07786 1B26 =FINDLB DO=(5) =PRGMST
           5F2
3244 0778D 146          C=DATO A
3245 07790 134          DO=C
3246 07793 184          DO=DO- 5              POINTS TO LABEL CHAIN HEAD IN PROGR
3247 07796 541          GONC   FDLB20         (B.E.T.)
3248 07799 181   FDLB10 DO=DO- 2              POINTS BACK TO BEGIN TOKEN
3249 0779C 3100         LC(2)  =tLBLST        SEE IF IS A LABEL TOKEN
3250 077A0 14A          A=DATO B
3251 077A3 161          DO=DO+ 2              POINTS TO LINK FIELD
3252 077A6 962          ?A=C   B              IS THIS A LABEL STATEMENT ?
3253 077A9 C1           GOYES  FDLB30         IF SO, LET'S TAKE A LOOK
3254 077AB 146   FDLB20 C=DATO A              READ THE LINK OF CURRENT LINE
3255 077AE 8AA          ?C=0   A              IS THE LINK = 0 ?
3256 077B1 00           RTNYES                IF SO, SAY NOT FOUND
3257 077B3 E6           C=C+1  A              REACHED ENDOF LINK ?
3258 077B5 400          RTNC                  IF SO, RETURN NOT FOUND, CARRY SET
3259 077B8 CE           C=C-1  A
3260 077BA 132          ADOEX
3261 077BD CA           A=A+C  A
3262 077BF 130          DO=A                  POINTS TO NEXT LINK
3263 077C2 56D          GONC   FDLB10         KEEP LOOKING  (B.E.T.)
3264 077C5 164   FDLB30 DO=DO+ 5              POINTS TO BEGIN OF LABEL
3265 077C8 7B10         GOSUB  LBLNAM         GET THE LABEL INTO REG-A
3266 077CC 184          DO=DO- 5              DO POINTS AT LABEL LINK FIELD
3267 077CF 974          ?A#B   W              DO WE HAVE A MATCH ?
3268 077D2 9D           GOYES  FDLB20         IF NOT, KEEP GOING
3269 077D4 185          DO=DO- 6              DO pts at @ or 2nd byte of line#
3270 077D7 14A  =ATCHK  A=DATO B              Read @ (F4) or MSD,MSD-1 of line#
3271 077DA F0           ASL    A              Shift High nib to XS field
3272 077DC B24          A=A+1  XS             Test for F --> @
3273 077DF 450          GOC    Rtncc*         Yes -- Return Clear Carry
3274 077E2 183          DO=DO- 4              DO POINTS AT EOL
3275 077E5 03    Rtncc* RTNCC
```

```
3276                    STITLE LBLNAM - Get Label Name into A
3277                ****************************************************************
3278                ****************************************************************
3279                **
3280                ** Name:(S) LBLNAM  -  Get Label Name into Register A
3281                **
3282                ** Category:   EXCUTL
3283                **
3284                ** Purpose:
3285                **      Get label name into Register A
3286                **
3287                ** Entry:
3288                **      D0      @  Beginning of Label in Memory
3289                **
3290                ** Exit:
3291                **      Carry clear
3292                **      P       =  0
3293                **      A       = Label name, Right justified with trailing
3294                **                blanks
3295                **
3296                **                "ABC" = 2020202020434241  (hex)
3297                **
3298                ** Calls:      BLANKC
3299                **
3300                ** Uses.......
3301                **  Exclusive: A,C,P
3302                **  Inclusive: A,C,P
3303                **
3304                ** Stk lvls:   1
3305                **
3306                **
3307                ** History:
3308                **
3309                **    Date      Programmer              Modification
3310                **  --------    ----------   ------------------------------------
3311                **  06/30/82    JP           Modified documentation
3312                **  10/08/82    JP           Added BLANKC call
3313                **
3314                ****************************************************************
3315                ****************************************************************
3316 077E7 7D20 =LBLNAM GOSUB  BLANKC          Load C with blanks
3317 077EB 21          P=     1
3318 077ED 1527        A=DAT0 W
3319 077F1 B04   LBNM10 A=A+1  P
3320 077F4 4C0         GOC    LBNM20
3321 077F7 0C          P=P+1                   P=15 ?
3322 077F9 490         GOC    LBNM30          yes
3323 077FC 0C          P=P+1
3324 077FE 52F         GONC   LBNM10
3325 07801 0D    LBNM20 P=P-1
3326 07803 0D    LBNM30 P=P-1
3327 07805 AFA         A=C    W               Move blanks to A
3328 07808 1521        A=DAT0 WP              Read label into A
3329 0780C 20          P=     0
3330 0780E 03          RTNCC
```

```
3331            ********************************************************************
3332            ********************************************************************
3333            **
3334            ** Name:    BLANKC  -  Load C with 8 blanks
3335            **
3336            ** Category:   GENUTL
3337            **
3338            ** Purpose:
3339            **      Load C with all blanks
3340            **
3341            ** Entry:
3342            **      2 entry points:
3343            **        1. BLNKC+ - Calls LDCSET: Sets D @ AVMEME, DO@OUTBS,
3344            **                    Saves DO in B(A), then ...
3345            **        2. BLANKC - Loads C with 8 blanks
3346            **
3347            ** Exit:
3348            **      P untouched
3349            **      Carry clear
3350            **      C(W)   =  All ASCII blanks
3351            **
3352            ** Calls:      BLNKC+: LDCSET
3353            **             BLANKC: None
3354            **
3355            ** Uses.......
3356            **  Exclusive: C
3357            **  Inclusive: B(A), C, D(A), DO (BLNKC+ entry)
3358            **
3359            ** Stk lvls:   BLNKC+: 2
3360            **             BLANKC: 0
3361            **
3362            ** History:
3363            **
3364            **    Date     Programmer             Modification
3365            **  --------   ----------   ----------------------------------
3366            **  10/08/82   JP           Added routine
3367            **  06/10/83   S.W.         Added BLNKC+ entry point to pack.
3368            **
3369            ********************************************************************
3370            ********************************************************************
3371 07810 8E00 =BLNKC+ GOSUBL =LDCSET
          00
3372 07816 D5          B=C    A
3373 07818 3F02 =BLANKC LCASC  \       \
          0202
          0202
          0202
          02
3374 0782A 03          RTNCC
```

```
3375                    STITLE FCHLBL - Find Label in Current File
3376                  ****************************************************************
3377                  ****************************************************************
3378                  **
3379                  ** Name:(S) FCHLBL  -  Find Label in Current BASIC File
3380                  **
3381                  ** Category:   FILUTL
3382                  **
3383                  ** Purpose:
3384                  **      Find a label in the current BASIC file
3385                  **      Assumes current file is BASIC
3386                  **
3387                  ** Entry:
3388                  **      Assumes current file is BASIC
3389                  **      R3      = Label to find
3390                  **                Right justified with trailing blanks
3391                  **
3392                  **      Falls into COMPL#
3393                  **
3394                  ** Exit:
3395                  **      P       =  0
3396                  **
3397                  **      Carry Clear - Label Found
3398                  **          D0 @ EOL preceding line containing Label
3399                  **          D1 @ Line # of line containing Label
3400                  **
3401                  **      Carry Set - Label Not Found in Current file
3402                  **
3403                  ** Calls:      GETSTC,TKSCN7,LBLNAM
3404                  **
3405                  ** Uses.......
3406                  **  Exclusive: A,B(A),C,D(A),D0,D1,R2,P
3407                  **  Inclusive: A,B(A),C,D(A),D0,D1,R2,P
3408                  **
3409                  ** Stk lvls:   2
3410                  **
3411                  ** Detail:
3412                  **      Fall into COMPL# to compute Line# after Label found
3413                  **      This code must IMMEDIATELY precede COMPL#
3414                  **
3415                  ** History:
3416                  **
3417                  **    Date      Programmer              Modification
3418                  **  --------    ----------    ---------------------------------
3419                  **  06/30/82    JP            Modified documentation
3420                  **
3421                  ****************************************************************
3422                  ****************************************************************
3423 0782C 76FE  =FCHLBL GOSUB  GETSTC          Get start/end of current file
3424 07830 3100   FCHLB2 LC(2)  =tLBLST         Label statement token
3425 07834 8E00          GOSUBL =TKSCN7         Look for Label statement
           00
3426 0783A 440           GOC    FCHLB4          Label found
3427 0783D 02            RTNSC                  Label not found
3428 0783F 181    FCHLB4 D0=D0- 2
```

```
3429 07842 132          ADOEX
3430 07845 D8           B=A    A              SAVE ADDR OF LINE LENGTH IN B
3431 07847 132          ADOEX
3432 0784A 168          DO=DO+ 4+5            DO POINTS TO BEGIN OF LABEL
3433 0784D 769F         GOSUB  LBLNAM         Move label to A
3434 07851 11B          C=R3                  Label to find
3435 07854 972          ?A=C   W              Match ?
3436 07857 41           GOYES  FCHLB6
3437 07859 D4           A=B    A
3438 0785B 130          DO=A                  Restore @ stmt length nibble
3439 0785E DO           A=0    A
3440 07860 14A          A=DATO B              Read statement length
3441 07863 CO           A=A+B  A              Next statement address
3442 07865 130          DO=A
3443 07868 57C          GONC   FCHLB2         Keep looking (B.E.T.)
3444            *
3445            * Label found
3446            *
3447 0786B D4    FCHLB6 A=B    A
3448 0786D 130          DO=A                  DO POINTS TO L.L OF THE STMT
3449            *
3450            *  START FROM HERE, FALL INTO COMPUTE LINE# ROUTINE
3451            *
```

```
3452                    STITLE COMPL# - Compute Line #
3453                ********************************************************************
3454                ********************************************************************
3455                **
3456                ** Name:     COMPL#  -  Compute Line # with D0 @ line length
3457                ** Name:(S) CPL#10  -  Compute Line # with D0 anywhere in stmt
3458                ** Name:     CPL#15  -  Compute Line # with C anywhere in stmt
3459                **
3460                ** Category:   FILUTL
3461                **
3462                ** Purpose:
3463                **      Compute Line # from position within statement
3464                **
3465                ** Entry:
3466                **      COMPL#:  D0 @ Line length of statement
3467                **      CPL#10:  D0 @ anywhere within statement
3468                **      CPL#15:  C  @ anywhere within statement
3469                **
3470                ** Exit:
3471                **      Carry Clear => Line # found
3472                **         P      =  0
3473                **         D1     @  Line #
3474                **         D0     @  EOL preceding the Line#  (D0=D1-2)
3475                **      Carry set => The input pointer is not pointing at
3476                **         current file.
3477                **
3478                ** Calls:      GETST-,
3479                **
3480                ** Uses.......
3481                **  Exclusive: A(A),B(A),C(A),D(A),D0,D1
3482                **  Inclusive: A(A),B(A),C(A),D(A),D0,D1
3483                **
3484                ** Stk lvls:   2
3485                **
3486                ** Note:  This routine will not check file type, it assumes the
3487                **        current file is type BASIC.
3488                **
3489                ** Detail:     Do not call this routine if specified address is
3490                **             at initial tEOL.  If at low nib of initial tEOL
3491                **             will return with carry set; if at high nib of
3492                **             initial tEOL will not work properly - found in
3493                **             code review (S.W.)
3494                **
3495                ** History:
3496                **
3497                **    Date      Programmer             Modification
3498                **  --------    ----------   ------------------------------------
3499                **  06/30/82    JP           Modified documentation
3500                **  01/04/83    JP           Removed S9 usage
3501                **  03/30/83    SC           Modified documentation
3502                **  04/15/83    JP           Fixed check for @ (F4)
3503                **
3504                ********************************************************************
3505                ********************************************************************
3506 07870 181    =COMPL# D0=D0- 2              BACK UP 2 NIBBLES
```

```
3507 07873 14A          A=DAT0 B
3508 07876 F0           ASL    A              Move high nib to A(XS)
3509 07878 B24          A=A+1  XS             Test for F ---> @
3510 0787B 4B0          GOC    CPL#10         IF SO, GOTO CMPL10
3511                    *
3512                    * IF NOT FOLLOWING AN @, MUST BE FOLLOWING A LINE #
3513                    *
3514 0787E 181          D0=D0- 2              D0 POINTS AT LINE#
3515 07881 136          CDOEX
3516 07884 5F4          GONC   CPL#70         (B.E.T.)
3517                    *
3518                    * D0 anywhere within line
3519                    *
3520 07887 136  =CPL#10 CDOEX
3521                    *
3522                    * C anywhere within line
3523                    *
3524                    * GETST- resets P to 0
3525                    *
3526 0788A D5   =CPL#15 B=C    A              SAVE ADDRESS IN B
3527 0788C 21           P=     1              Don't read filetype or use R2
3528 0788E 769E         GOSUB  GETST-         GET 1ST LINE OF FILE
3529 07892 161          D0=D0+ 2              D0 past the tF0
3530 07895 136          CDOEX                 COPY D0 TO C(A)
3531 07898 134          D0=C                  CHECK IF POINTING IN CURRENT FILE
3532 0789B 8B5          ?B<C   A
3533 0789E 00           RTNYES
3534 078A0 D9           C=B    A
3535 078A2 8BF          ?C>=D  A              D(A) IS @ END OF FILE
3536 078A5 00           RTNYES
3537 078A7 D0           A=0    A
3538 078A9 136  CPL#25  CDOEX
3539 078AC 8B5          ?C>B   A
3540 078AF 32           GOYES  CPL#60
3541 078B1 D7           D=C    A              SAVE CURRENT ADDR AS LAST ADDR
3542 078B3 134          D0=C
3543 078B6 163          D0=D0+ 4              PASS THE LINE #, POINTING AT L.L
3544 078B9 14A  CPL#30  A=DAT0 B
3545 078BC 136          CDOEX
3546 078BF C2           C=C+A  A              C= ADDR OF END OF STMT
3547 078C1 134          D0=C
3548 078C4 14A          A=DAT0 B
3549 078C7 161          D0=D0+ 2
3550 078CA 908          ?A=0   P              AT EOL ?
3551 078CD CD           GOYES  CPL#25         IF SO, GOTO CPL#25
3552 078CF 59E          GONC   CPL#30
3553 078D2 DB   CPL#60  C=D    A
3554 078D4 135  CPL#70  D1=C                  D1 POINTS TO LINE #
3555 078D7 134          D0=C
3556 078DA 181          D0=D0- 2              D0 points at EOL
3557 078DD 03   Rtncc-  RTNCC
```

```
3558                    STITLE PFINDL - Find Line# within program
3559                ****************************************************************
3560                ****************************************************************
3561                **
3562                ** Name:(S) PFINDL  -  Find Line# Within Program
3563                ** Name:(S) PFNDZL  -  Find Line# Within Program
3564                ** Name:    PFNDL*  -  Find Line# Within Program
3565                **
3566                ** Category:   FILUTL
3567                **
3568                ** Purpose:
3569                **      Find Line# between current program boundary
3570                **
3571                ** Entry:
3572                **      PFINDL:
3573                **        P = 0
3574                **        Assumes PRGMST, PRGMEN are current and updated
3575                **        D0 past Line# token
3576                **        Clears sXWORD (S9) flag to use Compiled Line# reference
3577                **      PFNDZL:
3578                **        P = 0
3579                **        Same entry as PFINDL
3580                **        D0 past Line# token
3581                **        Assumes sXWORD (S9) is set so:
3582                **          Will always search for Line#
3583                **          Allows XWORD entry, to search for Line# and not
3584                **          rely on compiled line# address, which may be bad.
3585                **      PFNDL*:
3586                **        P = 0
3587                **        D @ End of range to search for Line#
3588                **        D0 past Line# token
3589                **           Used by RENUMBER
3590                **
3591                ** Exit:
3592                **      P      =  0
3593                **      D0     =  D0 on entry  (past Line# token)
3594                **
3595                **      Carry Set - Line# found
3596                **          D1 @ Line#
3597                **          Reference to Line# (entry D0) is "compiled"
3598                **             Relative address to line# is filled in
3599                **
3600                **      Carry Clear - Line# not found
3601                **
3602                ** Calls:      SCOPCK,ISRAM?
3603                **
3604                ** Uses.......
3605                **  Exclusive: A(A),B(A),C(A),D(A),D1,D0,
3606                **             sXWORD (S9-PFINDL/PFNDZL only)
3607                **  Inclusive: A,B(A),C(A),D(A),D1,D0,
3608                **             sXWORD(S9-PFINDL/PFNDZL only)
3609                **
3610                ** Stk lvls:   2
3611                **
3612                ** NOTE:
```

```
3613            **      This routine will search between PRGMST & PRGMEN only
3614            **      if PFINDL or PFNDLZ is called.
3615            **
3616            **      PFNDZL will always search for Line# (if sXWORD set)
3617            **
3618            **      PFNDL* uses D(A) for boundary
3619            **
3620            **
3621            ** Detail:
3622            **      If not XWORD entry:
3623            **         It will look at the compiled address field following
3624            **            the line number first.
3625            **         If the compiled field is non-zero
3626            **            Compute the address of the Line#
3627            **      else
3628            **         Search the entire program
3629            **         Write the compiled address to RAM if Line# found
3630            **
3631            ** History:
3632            **
3633            **    Date      Programmer              Modification
3634            **  --------    ----------   ------------------------------------
3635            **  06/30/82    JP           Modified/Added Documentation
3636            **  02/04/83    JP           ISRAM? call uses all of A
3637            **  02/22/83    JP           Added PFNDZL entry added
3638            **  02/22/83    JP           Added S9 (sXWORD) usage
3639            **  03/08/83    JP           If not running; always search
3640            **
3641            ********************************************************************
3642            ********************************************************************
3643 078DF 849  =PFINDL ST=0   sXWORD          Clear XWORD entry flag
3644 078E2 163  =PFNDZL DO=DO+ 4               PASS ADDRESS FIELD
3645 078E5 146          C=DATO A               READ THE LINE #
3646 078E8 F6           CSR    A
3647 078EA D5           B=C    A               SAVE THE LINE # IN B
3648 078EC 183          DO=DO- 4               POINTS BACK TO ADDRESS FIELD
3649 078EF 879          ?ST=1  sXWORD          XWORD reference?
3650 078F2 92           GOYES  PFDL10          Don't check compiled field
3651 078F4 86D          ?ST=0  PgmRun          From keyboard?
3652 078F7 42           GOYES  PFDL10          Alway search for line#
3653 078F9 142          A=DATO A               READ THE COMPILED FIELD
3654 078FC 8A8          ?A=0   A               HAS THE LINE# COMPILED ?
3655 078FF C1           GOYES  PFDL10          IF NOT, GOTO FIND IT
3656 07901 136          CDOEX
3657 07904 CA           A=A+C  A               A POINTS TO THE LINE #
3658 07906 135          D1=C                   D1= DO AT ENTRY
3659 07909 75BD         GOSUB  =Scopck         VERIFY NOT JUMP OUT OF CURRENT PROG
3660 0790D 137          CD1EX
3661 07910 134          DO=C                   RESTORE DO TO ENTRY VALUE
3662 07913 131          D1=A                   D1 POINTS AT THE LINE #
3663 07916 46C          GOC    Rtncc-          CLEAR CARRY IF OUT OF RANGE
3664 07919 02           RTNSC                  RETURN WITH CARRY SET
3665            *
3666            * Set Program start and end
3667            *
```

```
3668 0791B 1F76  PFDL10 D1=(5) =PRGMEN
           5F2
3669 07922 147          C=DAT1 A
3670 07925 D7           D=C    A              D= PRGMEN
3671 07927 1C4          D1=D1- 5
3672 0792A 147          C=DAT1 A
3673 0792D D0           A=0    A
3674 0792F 5F1          GONC   PFDL65         (B.E.T.)
3675             *
3676             * Search program scope for line#
3677             *
3678 07932 D0   =PFNDL* A=0    A
3679 07934 D2    PFDL50 C=0    A
3680 07936 15F3         C=DAT1 4              C = LINE #
3681 0793A 8A1          ?B=C   A
3682 0793D 92           GOYES  PFDL70         FIND THE LINE #
3683 0793F 173          D1=D1+ 4              POINTS TO LINE LENGTH
3684 07942 14B          A=DAT1 B
3685 07945 137          CD1EX
3686 07948 8B7          ?C>D   A              PASSED PRGMEN YET ?
3687 0794B 36           GOYES  Rtncc+         IF SO, NOT FOUND, CLEAR CARRY
3688 0794D C2    PFDL60 C=C+A  A              POINTS TO END OF STMT
3689 0794F 135   PFDL65 D1=C
3690 07952 14F          C=DAT1 B
3691 07955 171          D1=D1+ 2
3692 07958 90A          ?C=0   P              AT EOL ?
3693 0795B 9D           GOYES  PFDL50
3694 0795D 14B          A=DAT1 B
3695 07960 137          CD1EX
3696 07963 59E          GONC   PFDL60
3697 07966 137   PFDL70 CD1EX                 CHECK IF PASS PRGMEN YET ?
3698 07969 135          D1=C
3699 0796C 8BF          ?C>=D  A
3700 0796F F3           GOYES  Rtncc+
3701             *
3702             * CHECK IF THE FILE IS WRITEABLE
3703             *
3704 07971 D7           D=C    A              SAVE D1 IN D(A)
3705 07973 8F00         GOSBVL =ISRAM?
           000
3706 0797A DB           C=D    A              RESTORE D1 FROM D(A)
3707 0797C 135          D1=C
3708 0797F D9           C=B    A              RESTORE C(A) FROM B(A)
3709 07981 5D0          GONC   PFDL75         IF NOT WRITEABLE DON'T WRITE TO IT
3710 07984 132          ADOEX                 COMPUTE THE OFFSET ANE WRITE IT
3711 07987 130          D0=A                   TO THE COMPILE FIELD
3712 0798A E2           C=C-A  A
3713 0798C 144          DAT0=C A
3714 0798F 02    PFDL75 RTNSC
3715             *
3716             * Clear Suspend  Annunciator/Flag
3717             *
3718 07991 311C =CLSUSP LC(2)  =flSUSP
3719 07995 6C4D         GOTO   SflagC         Clear SUSP flag/annunciator
```

```
3720                STITLE Null Program Check
3721            ********************************************************************
3722            ********************************************************************
3723            **
3724            ** Name:(S) NULLP   -  Null Program Check
3725            **
3726            ** Category:   FILUTL
3727            **
3728            ** Purpose:
3729            **      Check if current BASIC program is NULL
3730            **      Position to First line | EOF
3731            **
3732            ** Entry:
3733            **      File type will be checked if P=0
3734            **      If P=0: File Type is returned in R2 (from GETST-)
3735            **      If P#0: File Type will not be returned, nor checked.
3736            **
3737            **      Assumes: Length to data = Length to data of
3738            **               BASIC/ Binary file
3739            **      Assumes File type = BASIC or Binary or file with same
3740            **                          structure
3741            **
3742            ** Exit:
3743            **
3744            **      Carry Set   - Null program
3745            **      Carry Clear - not Null program
3746            **
3747            **        P  = 0
3748            **        D1 = First line of File (@ EOF)
3749            **        D = End of program
3750            **
3751            **        From GETST-
3752            **          A  = File Length
3753            **          R2 = File Type          (if P=0 on entry)
3754            **          D0 = First line of File
3755            **
3756            ** Calls:      GETST-
3757            **
3758            ** Uses.......
3759            **  Inclusive: A(A),C(A),P,D0,D1
3760            **  Exclusive: A(A),C(A),D(A),P,D0,D1,R2 (If P=0 on entry)
3761            **
3762            ** Stk lvls:   2
3763            **
3764            ** Detail:   Get start and end of current file (GETST-)
3765            **           Move First Line of file pointer to D1
3766            **           If file length = Offset to BASIC start of data
3767            **              RTNYES  (NULL program)
3768            **
3769            **           Null Program File Length = Offset BASIC start of dat
3770            **
3771            ** History:
3772            **
3773            **    Date     Programmer             Modification
3774            **  --------   ----------   ------------------------------------
```

```
3775            **  06/30/82   JP          Modified documentation
3776            **  01/04/83   JP          Remove S9 usage
3777            **  03/01/83   JP          Removed Hardwire Error Exit
3778            **
3779            ********************************************************************
3780            ********************************************************************
3781            *
3782 07999 7B8D =NULLP  GOSUB  GETST-        Get file start and end, Error if No
3783 0799D 136          CDOEX               First line of file
3784 079A0 135          D1=C                D1 = First line of file
3785 079A3 D2           C=0    A
3786 079A5 3111         LC(2)  oBSsod       Null BASIC program length
3787 079A9 8A2          ?A=C   A            NULL program ?
3788 079AC 00           RTNYES
3789 079AE 03    Rtncc+ RTNCC
```

```
3790                STITLE GOTO,GOSUB,RESTORE
3791           ********************************************************************
3792           ********************************************************************
3793           **
3794           ** Name:(S) GOTO    -  Statement Execution
3795           ** Name:(S) GOSUB   -  Statement Execution
3796           ** Name:    RESTOR  -  Statement Execution
3797           **
3798           ** Category:   STEXEC
3799           **
3800           ** Purpose:
3801           **      Execution of GOTO | GOSUB
3802           **      Partial execution of ON, ON ERROR, ON TIMER
3803           **      Partial execution of RESTORE
3804           **      Partial execution of XWORD with GOTO/GOSUB within
3805           **
3806           ** Entry:
3807           **
3808           **      GOSUB:  DO past GOSUB token       (Sets sGOSUB S3=1)
3809           **      GOTO:   DO past GOTO token        (Sets sGOSUB S3=0)
3810           **      RESTOR: DO past RESTORE token     (Sets S10=1)
3811           **              All status must be clear
3812           **      GOTO+ : Entry for statement containing:
3813           **              GOTO | GOSUB <lineno> | <label>
3814           **              DO @ <lineno> | <label> token past GOTO | GOSUB
3815           **              sEXTGS = 1 if External statement entry
3816           **              If GOSUB within statement
3817           **                 sGOSUB = 1     (S3)
3818           **              ON TIMER:    sONTMR = 1  (S6)
3819           **                           sEXTGS = 1
3820           **                           R3(S)  = Timer#
3821           **                           RSTK   = Return address
3822           **              ON ERROR:    sONERR = 1  (S4)
3823           **                           sEXTGS = 1
3824           **                           RSTK   = Return address
3825           **              ON RESTORE:  sRESTR = 1  (S10)
3826           **              External Entry:    (like ON INTRPT)
3827           **                           sEXTGS = 1
3828           **                           RSTK   = Return address
3829           **                           sGOSUB = 1 if GOSUB
3830           **              All other status MUST be clear!!!!
3831           **
3832           **
3833           **
3834           **              sXWORD = 1     (S9)
3835           **                 If XWORD with GOTO | GOSUB
3836           **                    Statement performing GOTO/GOSUB in  a
3837           **                    "sequential" fashion. EX: ON <exp> GOTO
3838           **
3839           **                 Guarantees always search for Line# referen
3840           **                 Eliminates problem of Line# reference address
3841           **                 that is invalid because it was not cleared
3842           **                 during PEDIT because the Lex File was missing.
3843           **
3844           **               External statements with GOTO/GOSUB that
```

```
3845            **              interupt program execute (ex:ON TIMER,ON INTR)
3846            **              must duplicate ONTIMR code (see JP&EXC) to
3847            **              guarantee proper TRACE of program execution.
3848            **              sXWORD must be set before jumping to GOTO+
3849            **
3850            **                R3(S) = Return type
3851            **                  If "normal" GOSUB
3852            **                     R3(S) = 0
3853            **                  If GOSUB from Keyboard (PgmRun=0)
3854            **                     R3(S) = 1
3855            **                  If "special" GOSUB/RETURN
3856            **                     See pRTNTp Poll in RETURN
3857            **                     R3(S) = 9 through 15
3858            **
3859            **                     This allows special processing when
3860            **                     RETURN of GOSUB is encountered
3861            **
3862            **              Assumes External Entry statements execute from a
3863            **              Program, i.e. PgmRun (S13) is set.
3864            **
3865            ** Exit:
3866            **      to BSCX60
3867            **         Avoids exception checking until AFTER the branch
3868            **         Cleans up TRACE
3869            **
3870            **      If RESTORE | ON RESTORE (sRESTR (S10))
3871            **         Jump to execute RESTORE
3872            **         Return to Run Loop thru NXTSTM
3873            **
3874            **      If RESTORE #
3875            **         Jump to execute RESTORE #
3876            **
3877            **      If GOTO from Keyboard
3878            **         Through NXTSTM after Setting CNTADR,CURRL
3879            **
3880            **      If Error (Label | Line# not found)
3881            **         If ON ERROR stmt          (sONERR (S7))
3882            **            Zero out ON ERROR address
3883            **         If ON TIMER, ON ERROR or External Entry
3884            **            PCADDR has been updated to ON statement
3885            **         If ON TIMER
3886            **            Appropriate Timer# has be OFFed
3887            **         Set up Error Message
3888            **         goto MFERR
3889            **
3890            ** Calls:       PFNDZL,FILXQT,FINDLB, PSHGSB,PRSCKB,
3891            **              TRFCK-,TRFROM,SNcrlf,TRTO+,CNTCUR,LNSKP-,
3892            **              SFGPGM,POPGSB,OFFTMR,CNTCK2,PSHUPD
3893            **
3894            ** Uses.......
3895            **  Exclusive: A,C,S0,S3,S4,S5,S6,S7,S8,S9,S10,S13,S14,R0-R2,D0,
3896            **              S-R0-0 (1 nib)
3897            **  Inclusive: A-D,S0-S10,S13,S14,R0-R4,D0,D1,all FUNCTION scrtch
3898            **              S-R1-0 thru S-R1-3,STMTD0,S-R0-0 (1nib)
3899            **
```

```
3900            **  PRSCKB uses R2; but its called only when NOT running
3901            **                  ON TIMER only active WHEN running
3902            **
3903            **  RSTK   = Return address          (If ON TIMER)
3904            **  R1     = Saved DO
3905            **  R3(S)  = Timer #                 (If sONTMR)
3906            **  R3(S)  = Return type             (If sXWORD)
3907            **  sGOSUB = GOSUB                   (S3)
3908            **  sONERR = ON ERROR entry          (S4)
3909            **  sEXTGS = External statement entry (S5)
3910            **  sONTMR = ON TIMER entry          (S6)
3911            **  sXWORD = XWORD entry for PFNDZL   (S9)
3912            **  sRESTR = RESTORE statement entry  (S10)
3913            **  PgmRun = Program running         (S13)
3914            **  NoCont = Don't Continue Run Loop  (S14)
3915            **  S-R0-0 = Timer#
3916            **
3917            ** Stk lvls:   7
3918            **
3919            ** Detail: (GOTO | GOSUB) (<lineno> | <label>)
3920            **         RESTORE [ <lineno> | <label> ]
3921            **         RESTORE # <assign#> [ <lineno> | <label> ]
3922            **         ON <exp> GOTO | GOSUB (<lineno>|<label>....)
3923            **         ON ERROR GOTO | GOSUB <lineno> | <label>
3924            **         ON TIMER # <exp>, <exp> GOTO | GOSUB <lineno>|<label>
3925            **
3926            **  RESTOR:  If next token = #
3927            **             go Execute RESTORE#   (RESTR#)
3928            **  RESTOR:  Set RESTORE flag        (sRESTR)
3929            **           If (no <line#> | <label>)
3930            **             Set C=0 (Indicates start of file for DATPTR)
3931            **             go Execute RESTORE    (RESTRX)
3932            **           goto GOTO+
3933            **  GOSUB:   Set GOSUB flag          (sGOSUB)
3934            **           goto GOTO-
3935            **  GOTO:    Clear GOSUB flag
3936            **  GOTO-:   Clear RESTORE, ON ERROR, External Entry flag
3937            **  GOTO+:   Save DO                          (R1)
3938            **           If not running
3939            **             Set program scope               (PRSCKB)
3940            **           Check if trace needed             (TRFCK,TRFROM)
3941            **           Restore Timer# to A(S)            (R3(S))
3942            **           If GOSUB
3943            **             Pop Return Address of Stack incase ON TIMER
3944            **             If not ON TIMER
3945            **               Calculate Return Address      (LNSKP-)
3946            **             If XWORD
3947            **               go Push Return type/addr      (goto 0)
3948            **             Set Return Type = 0
3949            **             If ON ERROR                     (sONERR)
3950            **               Save Return Address in ERRSUB to detect
3951            **               nesting of ON ERROR GOSUB statements
3952            **             If Return to keyboard (not PgmRun)
3953            **               Push CNTADR on GOSUB stack (PSHUPD)
3954            **               Return type = 1
```

```
3955            **              If ON TIMER
3956            **                 Shift Timer# to C(S)
3957            **                 Return type = Timer# + 1
3958            **       0:     Push Return type/addr on stack(PSHGSB)
3959            **           Save Timer# incase of Error      (S-R0-0)
3960            **           Restore D0                       (R1)
3961            **           If Line#
3962            **              Find line# address                (PFNDZL)
3963            **              If found
3964            **                 Position to EOF before Line#
3965            **                 Move Run address D1 -> C
3966            **
3967            **       1:        If RESTORE statement           (sRESTR)
3968            **    RESTRX:         Set DATPTR to C
3969            **                    golong NXTSTM
3970            **
3971            **                 If GOTO from Keyboard
3972            **                    Update CNTADR @ stmt jump& (CNTCUR)
3973            **                    Compute Line# of stmt jump &
3974            **                    Update CURRL @ Line#
3975            **                    golong to next stmt in Stmt Buffer
3976            **                 Set D0 @ Run/execution address (C)
3977            **                 Check if Trace Flow            (TRFCK-,TRTO+)
3978            **                 Restore D0 @ Execution address (R1)
3979            **                 Set PRGM annunc, PgmRun flag   (SFGPGM)
3980            **                 Goto to Run Loop
3981            **           If Label
3982            **              Move label into A                 (FILXQT)
3983            **              If Illegal Label or not in Current file
3984            **                 Error Exit                     (ERROR)
3985            **              else
3986            **                 Find label                     (FINDLB)
3987            **                 If label not found ---> Error exit
3988            **              Move Label stmt start (Run address)  D0 -> C
3989            **              goto 1;
3990            **   ERROR:   If line# or label not found
3991            **              If GOSUB
3992            **                 Pop Return address off stack   (POPGSB)
3993            **              If ON ERROR statement             (sONERR)
3994            **                 Clear ERRSUB address
3995            **                 Clear ON ERROR address
3996            **              If ON TIMER                    (sONTMR)
3997            **                 OFF appropriate Timer       (OFFTMR)
3998            **              If Trace mode --> Send CR/LF   (SNcrlf)
3999            **              Error Exit --> eSTMNF          (MFERR)
4000            **
4001            ** History:
4002            **
4003            **   Date      Programmer     Modification
4004            ** --------    -----------    ------------------------------
4005            ** 02/04/83    JP             Saving Timer# in scratch
4006            ** 02/07/83    JP             Add sXWORD status, PFNDLZ call
4007            ** 03/08/83    JP             Checking sEXTGS instead of sONTMR
4008            ** 03/31/83    JP             Remove UPDPC if External Entry
4009            ** 04/29/83    JP             If sXWORD, R3(S) = Return type
```

```
4010                ** 05/27/83    JP          If GOSUB from keyboard save CNTADR
4011                **                         on GOSUB stack
4012                **
4013                ** 06/17/83    JP          If GOTO from keyboard; set SUSP
4014                ** 06/29/83    JP          Check TRACE to before set PgmRun
4015                **                         Set PgmRun ALWAYS
4016                **
4017                ************************************************************
4018                ************************************************************
4019                *
4020 079B0 0000          REL(5) =RESTDC        Decompile Address for RESTORE
           0
4021 079B5 0000          REL(5) =RESTRP        Parse Address for RESTORE
           0
4022 079BA 14A  =RESTOR A=DAT0 B               Look for #,End Line or parm
4023                *
4024                * If RESTORE #
4025                *    go execute RESTORE#
4026                *
4027 079BD 3132          LCASC  \#\
4028 079C1 966           ?A#C   B              Not RESTORE # ?
4029 079C4 90            GOYES  RESTR1
4030 079C6 8D00          GOVLNG =RESTR#        go Execute RESTORE #
           000
4031                *
4032                * status clear on entry
4033                *
4034 079CD 85A   RESTR1 ST=1   sRESTR          Set RESTORE statement flag
4035 079D0 310F          LCHEX  F0             End of Line cut-off
4036 079D4 9E2           ?A<C   B              RESTORE with parameters ?
4037 079D7 92            GOYES  GT0001
4038                *
4039                * If RESTORE with no parameters
4040                *    Set C=0 to indicate start of file for DATPTR
4041                *    go execute RESTORE
4042                *
4043 079D9 D2            C=0    A              Start of file indicator
4044 079DB 62F0          GOTO   RESTRX         Execute RESTORE
4045                *
4046 079DF 0000          REL(5) =GOSBDC        Decompile Address for GOSUB
           0
4047 079E4 0000          REL(5) =GOSUBP        Parse Address for GOSUB
           0
4048                *
4049 079E9 853   =GOSUB  ST=1   sGOSUB         Set GOSUB flag
4050 079EC 6010          GOTO   GOTO-
4051                *
4052 079F0 0000          REL(5) =GOTODC        Decompile Address for GOTO
           0
4053 079F5 0000          REL(5) =GOTOP         Parse Address for GOTO
           0
4054                *
4055 079FA 843   =GOTO   ST=0   sGOSUB         Set GOTO flag/Clear GOSUB flag
4056 079FD 84A    GOTO-  ST=0   sRESTR         Clear RESTORE flag
4057 07A00 844    GT0001 ST=0   sONERR         Clear ON ERROR flag
```

```
4058 07A03 845          ST=0   sEXTGS          Clear External Entry flag
4059               *
4060               * EXTERNAL statement Entry
4061               *
4062 07A06 136  =GOTO+  CDOEX                  Save D0 in R1
4063 07A09 109         R1=C
4064               *
4065               * If not running
4066               *    Set Program start and end
4067               *    This must be done if Keyboard Execute
4068               * If Trace needed
4069               *    Trace FROM
4070               *
4071 07A0C 7871         GOSUB  PRSCKB          Set PRGMST & PRGMEN if not running
4072 07A10 7D31  GTO005 GOSUB  TrfCk-          CHECK IF NEED TO TRACE
4073 07A14 490          GOC    GTO010          No
4074 07A17 8F00         GOSBVL =TRFROM         Trace FROM
           000
4075               *
4076               * Restore Timer# to A(S)
4077               * If GOSUB
4078               *    If ON TIMER or External Statement
4079               *       RSTK = Next Statement address
4080               *    else
4081               *       Compute Next Statement Address
4082               *    A(A) = Return Address
4083               *
4084 07A1E 113   GTO010 A=R3                   Restore Timer# to A(S)
4085 07A21 863          ?ST=0  sGOSUB          not GOSUB ?
4086 07A24 16           GOYES  GTO060
4087 07A26 07           C=RSTK                 Pull Return Address incase
4088 07A28 DA           A=C    A               this is ON TIMER
4089 07A2A 875          ?ST=1  sEXTGS          ON TIMER or External Stmt?
4090 07A2D 80           GOYES  GTO020
4091 07A2F 8E00         GOSUBL =LNSKP-         Compute Next Statement address
           00
4092               *
4093               * If XWORD entry (sXWORD)
4094               *    R3(S) contains Return type
4095               *
4096 07A35 AC6   GTO020 C=A    S               Move possible return type to C(S)
4097 07A38 879          ?ST=1  sXWORD          XWORD entry ?
4098 07A3B 44           GOYES  GTO050          Return type already set
4099 07A3D AC2          C=0    S               Zero Return type nibble
4100               *
4101               * If ON ERROR  (sONERR)
4102               *    ERRSUB <-- Return address
4103               *    Return address is saved to detect ON ERROR GOSUB nesting
4104               *
4105 07A40 864          ?ST=0  sONERR          Not ON ERROR ?
4106 07A43 F0           GOYES  GTO030
4107 07A45 1F38         D1=(5) =ERRSUB         ON ERROR GOSUB Return address
           6F2
4108 07A4C 141          DAT1=A A
4109 07A4F 5F2          GONC   GTO050          B.E.T.
```

```
4110                *
4111                * If not Running
4112                *    Save CNTADR on stack
4113                *    Return type = 1   (Return to Keyboard)
4114                *
4115 07A52 87D   GT0030 ?ST=1  PgmRun         Running ?
4116 07A55 F1           GOYES  GT0040
4117 07A57 100          R0=A                  Save Return address
4118 07A5A 8EE9         GOSUBL CNTCK2         Read Continue address
          7F
4119 07A60 DA           A=C    A
4120 07A62 8E00         GOSUBL =PSHUPD        Push CNTADR on GOSUB stack
          00
4121 07A68 110          A=R0                  Restore Return address
4122 07A6B AC2          C=0    S
4123 07A6E B46          C=C+1  S              Return to Keyboard = 1
4124 07A71 5D0          GONC   GT0050         B.E.T.
4125                *
4126                * If ON TIMER
4127                *    Return type = Timer# + 1
4128                *
4129 07A74 866   GT0040 ?ST=0  sONTMR          Not ON TIMER ?
4130 07A77 80           GOYES  GT0050         Return to program = 0
4131 07A79 AC6          C=A    S              Move Timer# to C(S)
4132 07A7C B46          C=C+1  S              ON TIMER return = Timer# + 1
4133                *
4134                * Push Return address and Return Type onto GOSUB stack
4135                *
4136 07A7F 8E00  GT0050 GOSUBL =PSHGSB        Push address and type on stack
          00
4137                *
4138                * Save Timer# in scratch incase <lineno> or <label> not found
4139                *   So Timer# can be offed
4140                * Restore D0 from R1
4141                * Determine if <lineno> or <label> token
4142                *
4143 07A85 1B17   GT0060 D0=(5) =S-R0-0
          8F2
4144 07A8C 1504         DAT0=A S              Save Timer# in scratch
4145 07A90 119          C=R1
4146 07A93 134          D0=C                  Restore D0 within statement
4147 07A96 14A          A=DAT0 B              Read token
4148 07A99 161          D0=D0+ 2              Position to past token
4149 07A9C 3100         LC(2)  =tLINE#        Line # token
4150 07AA0 962          ?A=C   B              Line# token
4151 07AA3 91           GOYES  GT0110
4152                *
4153                * GOTO | GOSUB label
4154                *
4155                *    External Statement entry containing:
4156                *     GOTO|GOSUB <label>
4157                *       sEXTGS = 1    (External entry)
4158                *       sGOSUB = 1 if GOSUB
4159                *       D0     @ Label in RAM
4160                *
```

```
4161 07AA5 7298          GOSUB  FILXQT          GET THE LABEL INTO REG-A
4162 07AA9 526           GONC   GTOLE0          Label not good/not found
4163              *
4164              * Total label has been decompiled
4165              *       Find label
4166              *       If label not found ---> Error Exit
4167              *
4168 07AAC AF8           B=A    W               SAVE LABEL IN REG-B
4169 07AAF 73DC          GOSUB  FINDLB          Local GOTO/GOSUB/RESTORE
4170 07AB3 485           GOC    GTOLE0          LABEL NOT FOUND
4171 07AB6 136           CDOEX                  RUN address in C(A)
4172 07AB9 5F0           GONC   GTO120          B.E.T.
4173              *
4174              * Find Line#
4175              *    If XWORD
4176              *       Find line# by always searching for Line#
4177              *       sXWORD is set on entry to PFNDZL
4178              *    else
4179              *       Find line# using Line# reference address
4180              *    DO @ Compile address field of Line#
4181              *
4182 07ABC 722E  GTO110 GOSUB  PFNDZL          Find Line# using Line# reference
4183 07AC0 5B4           GONC   GTOLE0          Line# NOT found
4184              *
4185              * Line# found
4186              *
4187 07AC3 1C1           D1=D1- 2               Position to EOL before line#
4188 07AC6 137           CD1EX                  Move Run address to C
4189              *
4190              * Line number or Label found
4191              *   C = Start of statement containing Line# or Label
4192              *
4193              * If ON RESTORE | RESTORE statmenet  (sRESTR)
4194              *    Set DATPTR to C (start of DATA stmt scan)
4195              *    go execute Next Statement
4196              *
4197 07AC9 86A   GTO120 ?ST=0  sRESTR          Not RESTORE ?
4198 07ACC F0            GOYES  GTO130
4199              * Carry clear here
4200              *
4201 07ACE 1F29  RESTRX D1=(5) =DATPTR         Set Data Pointer
          6F2
4202 07AD5 145           DAT1=C A               at DATA stmt scan point
4203 07AD8 501           GONC   nxtstm          B.E.T.
4204              *
4205              * If GOTO from Keyboard
4206              *
4207              *    ALLOWS:  Setting Continue Address from Keyboard w/o STOP
4208              *
4209              *    Set SUSP annunciator
4210              *    Update Continue Address @ Run Address  (Stmt to GOTO)
4211              *    Calculate Line# of Stmt to GOTO
4212              *    Update Current Line to Line# of Stmt to GOTO
4213              *       Must position (C) into stmt line before Line# calculation
4214              *       C could be at EOL of statement before, causing problems
```

```
4215                *    go Position to "next" statement within Statement Buffer
4216                *       and Return to Run Loop
4217                *
4218 07ADB 873    GTO130 ?ST=1  sGOSUB         GOSUB ?
4219 07ADE F0            GOYES  GTO140
4220 07AE0 87D           ?ST=1  PgmRun        GOTO & Program Running ?
4221 07AE3 A0            GOYES  GTO140
4222 07AE5 7F98          GOSUB  CNTCUR        Update CNTADR,set SUSP, update CURR
4223 07AE9 6BEB   nxtstm GOTO   Nxtstm        Return to Run Loop @ Next stmt
4224                *
4225                * If Trace Mode
4226                *    Trace TO
4227                * Set DO @ Next statement to Execute
4228                *
4229 07AED 134    GTO140 DO=C                  Set DO @ stmt start
4230 07AF0 7D50          GOSUB  TrfCk-        Need to Trace ?
4231 07AF4 4F0           GOC    GTO150
4232 07AF7 8F00          GOSBVL =TRTO+
             000
4233 07AFE 119           C=R1                 Restore DO
4234 07B01 134           DO=C
4235                *
4236                * Set Program Running flag & PRGM Annunciator
4237                * Return to BASIC Loop @ RUNRT1
4238                *
4239 07B04 7748   GTO150 GOSUB  SFGPGM         Set PgmRun & PRGM Annunc
4240 07B08 6ED9          GOTO   RUNRT1        Return to BASIC loop
4241                *
4242                * Error Exit - Label | Line# not found
4243                *     If ON ERROR statement (S4) in Error
4244                * NOTE ****************************************
4245                *        ERRSUB and ERRADR must be ADJACENT
4246                *        Clear ERRSUB address
4247                *        Clear ON ERROR address - Avoids infinite loop
4248                *     If ON TIMER statement in Error
4249                *        Restore Timer # from Scratch
4250                *        OFF appropriate Timer  (Timer# in A(0))
4251                *        Avoids Infinite Loop:
4252                *          golong to MFERR
4253                *          return to RUNRT1, CKEXCP
4254                *          Timer will go off (if short interval)
4255                *          Jump to ON TIMER, only to Error again
4256                *     Go load Error Number & Error Exit
4257                *
4258 07B0C 863    GTOLE0 ?ST=0  sGOSUB         Not GOSUB ?
4259 07B0F 11            GOYES  GTOLE1
4260 07B11 8E00          GOSUBL =POPGSB       Pop Return address of GOSUB stack
             00
4261 07B17 ACB           C=D    S             Move Timer#+1 to C(S)
4262 07B1A ACA           A=C    S             Move Timer#+1 to A(S)
4263 07B1D A4C           A=A-1  S             Compute Timer#
4264 07B20 864    GTOLE1 ?ST=0  sONERR         Not ON ERROR statement?
4265 07B23 01            GOYES  GTOLE2
4266 07B25 AF2           C=0    W             Clear ERRSUB,ON ERROR addresses
4267 07B28 1F38          D1=(5) =ERRSUB
```

```
            6F2
   4268 07B2F 15D9          DAT1=C 10
   4269 07B33 866    GTOLE2 ?ST=0  sONTMR          Not ON TIMER ?
   4270 07B36 11            GOYES  GTOLE3
   4271 07B38 1B17          D0=(5) =S-R0-0        Restore Timer# to A(0)
            8F2
   4272 07B3F 15A0          A=DAT0 1
   4273 07B43 7000          GOSUB  =OFFTMR        go OFF Timer# to avoid infinite loo
   4274               *
   4275               * If TRACE mode
   4276               *    Send CR/LF
   4277               *
   4278 07B47 8E00   GTOLE3 GOSUBL =SNcrlf        If in trace mode, send CR/LF
            00
   4279 07B4D 67A8   GTOLE4 GOTO   FTCER2         Label not found
   4280               *
   4281 07B51 8D00 =TrfCk- GOVLNG =TRFCK-
            000
```

```
4282                    STITLE UPDPC - Update PC address near D0
4283                ********************************************************************
4284                ********************************************************************
4285                **
4286                ** Name:     UPDPC   -  Update PCADDR, by Referencing D0
4287                ** Name:     UPDPCC  -  Update PCADDR from C
4288                **
4289                ** Category:   EXCUTL
4290                **
4291                ** Purpose:
4292                **       Updates PCADDR to point to the line length byte which
4293                **       follows the line# of the line in which D0 resides
4294                **
4295                ** Entry:
4296                **       UPDPC:  D0 points within stmt to set PCADDR at
4297                **       UPDPCC: C(A) holds value to set PCADDR at
4298                **
4299                ** Exit:
4300                **       UPDPC:
4301                **         Carry set:
4302                **           D0 not within Program scope
4303                **           PCADDR not updated
4304                **         Carry clear:
4305                **           P=0
4306                **           D1 @ PCADDR
4307                **           D0 @ tEOL prior to line
4308                **           C(A) contains new PCADDR
4309                **           PCADDR updated
4310                **
4311                **       UPDPCC:
4312                **         D1 @ PCADDR
4313                **         Carry preserved
4314                **         PCADDR updated from C(A)
4315                **
4316                ** Calls:       CPL#10 (UPDPC only)
4317                **
4318                ** Uses.......
4319                **  Exclusive: C(A),D1,PCADDR
4320                **  Inclusive: UPDPC only: A(A),B(A),C(A),D(A),D0,D1,PCADDR
4321                **
4322                ** Stk lvls:   UPDPC:  3
4323                **              UPDPCC: 0
4324                **
4325                ** Algorithm:
4326                **       Compute line# in which D0 resides (CPL#10)
4327                **       If not within Program scope
4328                **          Return carry
4329                **       Position to Line Length field
4330                **       Update PCADDR at Line Length field
4331                **
4332                ** History:
4333                **
4334                ** Date         Programmer      Modification
4335                ** --------     --------------  --------------------------------
4336                ** 06/27/83     JP              Check carry on rtn from CPL#10
```

```
4337                **
4338                ********************************************************************
4339                ********************************************************************
4340 07B58 7B2D =UPDPC  GOSUB  =CPL#10        Compute line # of statement
4341 07B5C 400          RTNC                 Not within Program scope
4342 07B5F 173          D1=D1+ 4             Position to Line length
4343 07B62 137          CD1EX
4344 07B65 1F97 =UPDPCC D1=(5) =PCADDR       Update PCADDR to point here
           6F2
4345 07B6C 145          DAT1=C A
4346 07B6F 01           RTN
```

```
4347                    STITLE PRSCOP - Compute Program Scope
4348                ****************************************************************
4349                ****************************************************************
4350                **
4351                ** Name:    PRSCOP  -  Compute Program Scope
4352                ** Name:    PRSCKB  -  Compute Program Scope; Return if SUSP
4353                ** Name:(S) PRSCOO  -  Compute Program Scope; GETSTC exit cond
4354                **
4355                ** Category:   EXCUTL
4356                **
4357                ** Purpose:
4358                **   Compute Program Scope:Program Start,Program End,Sub Links
4359                **
4360                ** Entry:
4361                **      Assumes: CURRST, CURREN pointing at current file
4362                **
4363                **      PRSCKB:  If program suspened  --> Return
4364                **               P=0
4365                **      PRSCOP:  Calls GETSTC to position in file and
4366                **                 check file type
4367                **               Error Exits if non-BASIC file
4368                **      PRSCOO:  Assumes positioning = GETSTC exit conditions
4369                **               File type must be BASIC;Binary or Same structure
4370                **               P=0
4371                **      PRSCO-:  Get program start/end w/o File Type error exit
4372                **               Allows Program scope set for Binary programs
4373                **      PRSC60:  Set Program Start and End only
4374                **               D1 @ PRGMST
4375                **
4376                ** Exit:
4377                **      If program already running on entry:
4378                **         This routine does nothing
4379                **
4380                **      If PRSCOP entry:
4381                **         If current file not BASIC
4382                **            Error Exit ---> eFTYPE
4383                **
4384                **      A   = PRGMST  (Program Start)
4385                **      C,D = PRGMEN  (Program End)
4386                **      D1  @ PRGMST
4387                **
4388                **
4389                ** Calls:      GETSTC,GETSTe,CHAIN*,RUSUS?,SCOPEN
4390                **
4391                ** Uses.......
4392                **  Exclusive: A(A),B(A),C(A),D(A),D0,D1,R2
4393                **  Inclusive: A,B(A),B(S),C,D(A),D0,D1,R2
4394                **
4395                ** Stk lvls:   3
4396                **
4397                ** NOTE:
4398                **   PRSCKB will not set program scope if running or suspended
4399                **   PRSCOP will always set the program scope if program not
4400                **     running
4401                **
```

```
4402                ** History:
4403                **
4404                **    Date     Programmer          Modification
4405                **  --------   ----------  ----------------------------------
4406                **  06/30/82   JP          Modified documentation
4407                **  09/15/82   JP          Changed GETSTC to error return
4408                **  01/04/83   JP          Added PRSCKB entry point
4409                **  02/11/83   JP          Deleted PRSC55 entry point
4410                **
4411                ************************************************************
4412                ************************************************************
4413                *
4414 07B71 87D    =RUSUS? ?ST=1  PgmRun
4415 07B74 00             RTNYES
4416                *
4417 07B76 311C           LC(2)  =flSUSP
4418 07B7A 8C00 =SFlag? GOLONG =sflag?
          00
4419                *
4420                *
4421 07B80 72AB =PRSC0- GOSUB  GETSTC       Get prog start w/o File type error
4422 07B84 6E00         GOTO   PRSC00
4423                *
4424                *
4425 07B88 75EF =PRSCKB GOSUB  RUSUS?       Running or suspended ?
4426 07B8C 400          RTNC                 If so, return
4427                *
4428 07B8F 7CCB =PRSCOP GOSUB  GETSTe        MAKE SURE THIS IS A BASCI FILE
4429 07B93 136  =PRSC00 CDOEX
4430 07B96 D5           B=C    A            B= ADDR OF THE 1ST EOL OF PRGM
4431 07B98 135          D1=C
4432 07B9B 1C9          D1=D1- 10           D1 POINTS AT SUB CHAIN HEAD
4433 07B9E 143          A=DAT1 A
4434 07BA1 8A8          ?A=0   A            THE FILE CHAINED YET ?
4435 07BA4 E3           GOYES  PRSC50       IF NOT, LET'S CHAIN IT
4436 07BA6 E4           A=A+1  A
4437 07BA8 4C1          GOC    PRSC10       SUBLNK not ALREADY ESTABLISHED
4438                *
4439                * Sublink already established
4440                * B(A) = Address of the first EOL of the program
4441                * A(A) = SUB link +1
4442                * D1 @ SUB link
4443                * Now I want to points at the token of the next SUB link
4444                *
4445 07BAB 1C2          D1=D1- 3
4446 07BAE 137          CD1EX
4447 07BB1 C2           C=A+C  A            C(A) @ two nibs preced next SUB link
4448 07BB3 135          D1=C
4449                *
4450 07BB6 7030         GOSUB  SCOPEN       See if it is a tSUB or tENDSUB
4451                *
4452 07BBA 137          CD1EX               Put program end to D(A)
4453 07BBD D7           D=C    A
4454 07BBF D4           A=B    A            A(A) = Program start
4455 07BC1 6900         GOTO   PRSC20
```

```
4456                *
4457                * SUBLNK = FFFFF, NO SUB-PROGRAM IN THIS FILE
4458                *
4459 07BC5 179        PRSC10 D1=D1+ 10            D1 POINTS TO 1ST EOL
4460 07BC8 133               AD1EX                A= PROGRAM START
4461 07BCB 1F26      =PRSC20 D1=(5) =PRGMST
           5F2
4462 07BD2 DB                C=D    A             C= PROGRAM END
4463                *
4464                *
4465 07BD4 141       =PRSC60 DAT1=A A             SET PRGMST
4466 07BD7 174               D1=D1+ 5
4467 07BDA 145        DT1=C- DAT1=C A             SET PRGMEN
4468 07BDD 1C4               D1=D1- 5
4469 07BE0 03                RTNCC
4470                *
4471                *SUBLNK= 0, SO WE HAVE TO COMPUTE IT
4472                *
4473 07BE2 7B40       PRSC50 GOSUB  CHAIN*
4474 07BE6 68AF              GOTO   PRSCOP        START IT ALL OVER AGAIN
4475                ********************************************************************
4476                ********************************************************************
4477                **
4478                ** Name:    SCOPEN  -  Define the Program Scope End
4479                **
4480                ** Category:   FILUTL
4481                **
4482                ** Purpose:
4483                **      When the end of a program or subprogram is a SUB or
4484                **      ENDSUB statement, determine the end of the scope.
4485                **
4486                ** Entry:
4487                **      P      =  0
4488                **      D1 @ the token tSUB or tENDSUB
4489                **
4490                **
4491                ** Exit:
4492                **      P      =  0
4493                **      D1 @ end of program scope
4494                **      Carry clear
4495                **
4496                ** Calls:      None
4497                **
4498                ** Uses.......
4499                **  Exclusive: A(B),C(B),D1
4500                **  Inclusive: A(B),C(B),D1
4501                **
4502                ** Stk lvls:   0
4503                **
4504                **
4505                **    Date     Programmer             Modification
4506                **  --------   ----------   ----------------------------------
4507                **  05/18/83   SC           Wrote
4508                **
4509                ********************************************************************
```

```
4510            ********************************************************************
4511            *
4512 07BEA 14B  =SCOPEN A=DAT1 B              Read the token
4513 07BED 3100         LC(2)  =tSUB
4514 07BF1 962          ?A=C   B
4515 07BF4 70           GOYES  SPEN10
4516            *
4517            * D1 @ tENDSUB
4518            *
4519 07BF6 178          D1=D1+ 9
4520 07BF9 03           RTNCC
4521            *
4522 07BFB 1C3   SPEN10 D1=D1- 4              D1 @ line # or "@"
4523 07BFE 14B          A=DAT1 B
4524 07C01 BE4          ASR    B
4525 07C04 B04          A=A+1  P
4526 07C07 550          GONC   SPEN20
4527            *
4528            * The SUB statement is proceeded by the "@" sign.
4529            * Scope end should be set to after the "@" sign.
4530            *
4531 07C0A 173          D1=D1+ 4
4532            *
4533            * The SUB statement is proceeded by a line number
4534            * Scope end should be set to right before the line number
4535            *
4536 07C0D 1C1   SPEN20 D1=D1- 2
4537 07C10 03           RTNCC
4538            *
4539            *
```

```
4540                    $TITLE CHAIN*- Chain sub-programs
4541                ********************************************************************
4542                ********************************************************************
4543                **
4544                ** Name:(S) CHAIN+  -  Chain Subprograms, Labels, DEF FNs
4545                ** Name:(S) CHAIN-  -  Chain Subprograms, Labels, DEF FNs
4546                ** Name:    CHAIN*  -  Chain Subprograms, Labels, DEF FNs
4547                **
4548                ** Category:   FILUTL
4549                **
4550                ** Purpose:
4551                **      Chain all Sub-programs in a file
4552                **      Chain all Labels in a file
4553                **      Chain all Def FNs in a file
4554                **
4555                ** Entry:
4556                **      P      = 0
4557                **      Assumes Current file is BASIC
4558                **
4559                **      CHAIN+:   Chain Current File
4560                **      CHAIN-:   A @ Start of file to chain
4561                **      CHAIN*:   D1 @ Sub-link of file
4562                **                D  @ End of file
4563                **
4564                ** Exit:
4565                **      P      = 0
4566                **      D(A) = End of file
4567                **      D1   @ Sub-link of file
4568                **
4569                **      Error Exit - if file not in RAM
4570                **                   eFACCS - " Illegal Access"
4571                **
4572                ** Calls:      FNDDO+ (FINDA), ISRAM?
4573                **
4574                ** Uses.......
4575                **  Exclusive: A,B(A),B(S),C,D(A),D0,D1,R2
4576                **             CHAIN*: D1 is preserved
4577                **
4578                ** Stk lvls:   2
4579                **
4580                ** History:
4581                **
4582                **    Date      Programmer              Modification
4583                **  --------    ----------   ----------------------------------
4584                **  06/30/82    JP           Modified documentation
4585                **  01/17/83    S.W.         Updated/Expanded documentation
4586                **
4587                ********************************************************************
4588                ********************************************************************
4589 07C12 1FD5 =CHAIN+ D1=(5) =CURRST
           5F2
4590 07C19 143          A=DAT1 A
4591 07C1C D2   =CHAIN- C=0    A
4592 07C1E 3102         LC(2)  =oFLENh
4593 07C22 CA           A=A+C  A
```

```
4594 07C24 131          D1=A                D1 @ FILE CHAIN LENGTH
4595 07C27 147          C=DAT1 A            C = FILE CHAIN LENGTH
4596 07C2A C2           C=A+C  A            C @ END OF FILE
4597 07C2C D7           D=C    A            D @ END OF FILE
4598 07C2E 174          D1=D1+ 5            D1 @ SUB CHAIN HEAD
4599 07C31 137  =CHAIN* CD1EX
4600 07C34 10A          R2=C                R2 POINTS TO LAST SUBLNK
4601 07C37 134          D0=C
4602 07C3A 135          D1=C
4603 07C3D 147          C=DAT1 A
4604 07C40 8AE          ?C#0   A            IS THE FILE AREADY CHAINED
4605 07C43 00           RTNYES              IF SO, JUST RETURN
4606 07C45 137          CD1EX               SEE IF THIS FILE IN SRAM OR IRAM
4607 07C48 8F00         GOSBVL =ISRAM?
          000
4608 07C4F D9           C=B    A
4609 07C51 135          D1=C
4610 07C54 480          GOC    CHN110
4611 07C57 8C00         GOLONG =PRGROM       Give eFACCS error
          00
4612            *
4613 07C5D 169   CHN110 D0=D0+ 10           D0 POINTS TO 1ST EOL
4614 07C60 174          D1=D1+ 5            D1 POINTS TO 1ST LBLLNK
4615 07C63 AC1          B=0    S            INITIALIZE ENDSUB FLAG
4616 07C66 D0    CHN120 A=0    A
4617 07C68 14A          A=DAT0 B
4618 07C6B 161          D0=D0+ 2
4619 07C6E 90C          ?A#0   P            ARE WE AT EOL NOW ?
4620 07C71 50           GOYES  CHN130       IF NOT, GOTO 130
4621 07C73 163          D0=D0+ 4            POINTS TO LINE LENGTH
4622 07C76 136   CHN130 CD0EX
4623 07C79 8BF          ?C>=D  A            REACHED END OF FILE YET ?
4624 07C7C B3           GOYES  CHN300       IF SO, GOTO CHN300
4625            **
4626 07C7E 136   CHN140 CD0EX
4627 07C81 14A          A=DAT0 B
4628 07C84 D8           B=A    A            B= LINE LENGTH
4629 07C86 8E00         GOSUBL =FNDD0+
          00
4630            *
4631 07C8C 00           CON(2) =tLBLST
4632 07C8E B40          REL(3) CHN510
4633            *
4634 07C91 00           CON(2) =tDEF
4635 07C93 640          REL(3) CHN510
4636            *
4637 07C96 00           CON(2) =tENDDF       C(B)= tENDDF
4638 07C98 140          REL(3) CHN510
4639            *
4640 07C9B 00           CON(2) =tSUB
4641 07C9D C20          REL(3) CHN250
4642            *
4643 07CA0 00           CON(2) =tENDSB       C(B)= tENDSB
4644 07CA2 920          REL(3) CHN280
4645            *
```

```
4646 07CA5 00          CON(2) 0
4647              *
4648              *
4649              *
4650 07CA7 132  CHN180 ADOEX                 SKIP OVER THE LINE
4651 07CAA C0          A=A+B  A
4652 07CAC 20   CHN190 P=     0
4653 07CAE 132         ADOEX
4654 07CB1 181         D0=D0- 2              Adjust D0 to @ or tEOL
4655 07CB4 51B         GONC   CHN120         GOTO SEE NEXT LINE (B.E.T.)
4656              *
4657 07CB7 D0   CHN300 A=0    A
4658 07CB9 CC          A=A-1  A
4659 07CBB 141         DAT1=A A              WRITE FFFFF TO LAST LBL.LINK
4660 07CBE 11A         C=R2
4661 07CC1 135         D1=C
4662 07CC4 141         DAT1=A A              WRITE FFFFF TO LAST SUB.LINK
4663 07CC7 03          RTNCC
4664              *
4665 07CC9 21   CHN250 P=     1
4666 07CCB 0C   CHN280 P=P+1
4667 07CCD 137         CD1EX                 RESTORE R2 & D1
4668 07CD0 12A         CR2EX                 Temp. save ptr to last LBLLNK
4669 07CD3 135         D1=C                  D1 POINTS TO LAST SUBLNK
4670 07CD6 AC1         B=0    S              Clear ENDSUB flag
4671 07CD9 161  CHN510 D0=D0+ 2
4672 07CDC 132         ADOEX
4673 07CDF 130         D0=A                  Ptr to next LNK field in A,D0
4674 07CE2 137         CD1EX
4675 07CE5 135         D1=C                  Address of last LNK field
4676 07CE8 EE          C=A-C  A              Calculate offset
4677 07CEA 94D         ?B#0   S              Is last one an ENDSUB
4678 07CED 50          GOYES  CHN520         If so, don't write it
4679 07CEF 145         DAT1=C A              Write out offset
4680 07CF2 131  CHN520 D1=A                  Update LNK pointer
4681 07CF5 181         D0=D0- 2
4682 07CF8 890         ?P=    0              Label Chain?
4683 07CFB CA          GOYES  CHN180         D1 POINTS PAST EOL
4684 07CFD 137         CD1EX                 RESTORE R2 & D1
4685 07D00 12A         CR2EX                 R2 is again ptr to last SUBLNK
4686 07D03 135         D1=C                  D1 is ptr to last LBLLNK
4687 07D06 D2          C=0    A
4688 07D08 CE          C=C-1  A
4689 07D0A 145         DAT1=C A              WRITE FFFFF TO LAST LABEL LINK
4690 07D0D 891         ?P=    1              END SUB?
4691 07D10 01          GOYES  CHN530         If so, set ENDSUB flag in B(S)
4692 07D12 132         ADOEX                 Program scope ends with SUB stmt
4693 07D15 C0          A=A+B  A              Start new label chain
4694 07D17 131         D1=A
4695 07D1A 1C6         D1=D1- 7
4696 07D1D 5E8         GONC   CHN190          (B.E.T.)
4697              *
4698 07D20 AC1  CHN530 B=0    S
4699 07D23 B45         B=B+1  S
4700 07D26 508         GONC   CHN180          (B.E.T.)
```

```
4701            *
```

```
4702                    STITLE CLRSTK   - Collapse Stacks/Zero addresses
4703            ****************************************************************
4704            ****************************************************************
4705            **
4706            ** Name:    CLRSTK  -  Clear Stack/Zero Addresses
4707            ** Name:    CLPSTK  -  Clear Stack/Zero Addresses
4708            ** Name:    ZERPGM  -  Zero program addresses
4709            **
4710            ** Category:   EXCUTL
4711            **
4712            ** Purpose:
4713            **      CLPSTK:
4714            **        Collapse all FOR/GOSUB/MATH stacks, pending Func stack
4715            **        Zero program addresses
4716            **        Used by RUN,EDIT,PEDIT,END ALL
4717            **      CLRSTK:
4718            **        Collapse FOR/GOSUB/MATH stacks to current prgm level
4719            **        Zero program addresses
4720            **        Used by END
4721            **      ZERPGM:
4722            **        Zero Program Addresses, Clear SUSP annunciator
4723            **        Used by MERGE, PURGE, KBRTCK (return to keyboard)
4724            **
4725            **      Zero Addresses:
4726            **        CONTinue, ON ERROR, ON ERROR GOSUB, ON INTERRUPT,
4727            **        DATA statement ptr, 3 ON TIMER addresses
4728            **      Zero
4729            **        3 Alarm RAM locations for 3 Timers
4730            **        Pending alarm bits for 3 Timers
4731            **      Clear SUSP Annunciator/Flag
4732            **
4733            **      Fast Poll to allow future statements to zero addresses/RA
4734            ** Entry:
4735            **      P       =  0
4736            **      See Purpose
4737            **
4738            ** Exit:
4739            **      P       =  0
4740            **      Carry clear
4741            **
4742            ** Calls:       C=RAME,RSTOPT,RBLDCH,PUTALM,INITPT,FPOLL,DT1=C-
4743            **              CLRSTK,CLSUSP,MANSTK
4744            **
4745            ** Uses.......
4746            **  Inclusive: A,B,C,D,D0,D1,ALRM(+36: ALRM1,2,3),PNDALM (+1)
4747            **              ANNAD1-4,flSUSP(-63),CNTADR thru TMRAD3,
4748            **              MTHSTK thru GSBSTK,PRMPTR
4749            **
4750            **          R0,R2 (CLPSTK only)
4751            **          R4 must be not be used due to call by CONFIGURATION
4752            **
4753            ** Stk lvls:   3
4754            **
4755            ** NOTE:
4756            **
```

```
4757            ** Detail:
4758            **
4759            **   CLPSTK :
4760            **      1. Collapes all local variable created by sub-program cal
4761            **         to leave only the calculator variables.
4762            **      2. Call CLRSTK.
4763            **
4764            **   CLRSTK :
4765            **      1. Set MTHSTK,FORSTK,GSBSTK = ACTIVE(collaped)
4766            **      2. Collapes stack generated by user defineded function.
4767            **      3. Zero CNTADR,ERRSUB,ERRADR,ONINTR,DATPTR,TMRAD1-3
4768            **      4. Clear SUSP annunciator
4769            **      5. Zero alarm RAM locations for all 3 Timers
4770            **
4771            ** History:
4772            **
4773            **    Date     Programmer            Modification
4774            **  --------   ----------   ----------------------------------
4775            **  06/30/82   JP           Modified documentation
4776            **  07/20/82   JP           Modified documentation
4777            **  07/21/82   SC           Change routine name
4778            **  01/19/83   JP           Added ZERPGM entry
4779            **  02/04/83   JP           Added pZERPG Fast Poll
4780            **  03/15/83   JP           Added DT1=C1; Collapse MTHSTK
4781            **  05/13/83   JP           Updated Uses documentation
4782            **  05/18/83   JP           D1=(5) CNTADR @ ZERPGM
4783            **  07/22/83   JP           R4 cannot be used due to CONFIG
4784            **
4785            ****************************************************************
4786            ****************************************************************
4787            ****************************************************************
4788            ****************************************************************
4789            **
4790            ** Name:(S) pZERPG  -  Poll to zero program information
4791            **
4792            ** Category:   POLL
4793            **
4794            ** Type:       FPOLL
4795            **
4796            ** Purpose:
4797            **      Fast poll to allow future statements to zero addresses
4798            **      and RAM associated with extending a statement, adding
4799            **      a statement or application.
4800            **
4801            **      This poll issued when zero program information due to
4802            **      an END, ENDALL, EDIT, Program Edit....
4803            **
4804            **      Issued from CLRSTK/CLPSTK/ZERPGM routine.
4805            **
4806            ** Should poll be "Handled" (return with XM=0)?:
4807            **      No - This poll should continue to ALL Lex files
4808            **
4809            ** Meaning of "Handling" Poll (what does code do if handled?):
4810            **      Zero appropriate RAM / addresses associated with
4811            **      statement or application.
```

```
4812            **
4813            ** Entry conditions for handler:
4814            **      Carry set on entry
4815            **      B[A] = Poll number (pZERPG)
4816            **      HEX mode.
4817            **      P=0.
4818            **
4819            **      BASIC stacks have been collapsed to appropriate
4820            **        level.
4821            **      CONT, ON ERROR, ON ERROR GOSUB, ON INTR, ON TIMER
4822            **        statement addresses have been zeroed
4823            **      Timer alarm RAM has be zeroed
4824            **      SUSP annunciator/flag has be cleared
4825            **
4826            ** Normal exit conditions from handler if handled (ST, RAM,
4827            ** registers, etc.):
4828            **      This poll should never be "handled".
4829            **      Always return with XM=1
4830            **
4831            ** Normal exit conditions from handler if not handled (ST, RAM,
4832            ** registers, etc.):
4833            **      HEX mode.
4834            **      XM=1.
4835            **      R registers intact. Status intact
4836            **
4837            ** Available subroutine levels:
4838            **      --FPOLL handler is two levels deeper than caller--
4839            **      The invoking routine (CLRSTK/CLPSTK/ZERPGM) uses 3 lvls
4840            **      Therefore, a handler may use ONLY 1 lvl.
4841            **
4842            **      Use RSTK<R to save 3 levels in RSTKBF circular buffer
4843            **      Use R<RSTK to restore 3 levels
4844            **
4845            ** What registers/RAM may be used if not handled?:
4846            **      --A-C, D[15-5] D0, D1, P always available (FPOLL only)--
4847            **      --NOTE: D[A] is sacred in FPOLL!!--
4848            **      Do not use an R registers, please !!!!
4849            **      Do not use Status
4850            **      Do not use S-R0-0
4851            **
4852            ** Envisioned application(s):
4853            **      Extend or add a statement (like ON INTP) and need
4854            **      to zero the RAM address associated with the statement.
4855            **
4856            **      Zero I/O Buffer associated with an application because
4857            **      all other program information is being zeroed.
4858            **
4859            ** Note:
4860            **      Do not use S-R0-0 under ANY circumstances.
4861            **       (counted on by PURGE ALL)
4862            **
4863            ** History:
4864            **
4865            **    Date     Programmer             Modification
4866            **  --------   ----------   ----------------------------------
```

```
4867                ** 02/04/83   JP          Added poll
4868                ** 04/23/83   JP          Revised/updated documentation
4869                ** 05/13/83   JP          Changed Usage documentation
4870                **
4871                ****************************************************************
4872                ****************************************************************
4873                *CLPSTK D1=(5) =RAMEND
4874                *       C=DAT1 A            C= RAMEND
4875 07D29 8F00 =CLPSTK GOSBVL =C=RAME          C= RAMEND
           000
4876 07D30 1C4          D1=D1- 5                D1 POINTS TO CALSTK
4877 07D33 143          A=DAT1 A
4878 07D36 8A6          ?A#C   A                ARE WE IN MIDDLE OF SUB ?
4879 07D39 85           GOYES  CLAP20           IF SO, go collapse each level
4880                *
4881                * SET GSBSTK,FORSTK,MTHSTK TO EQUAL TO ACTIVE
4882                *
4883 07D3B 1F8A =CLRSTK D1=(5) =ACTIVE          D1 POINTS TO ACTIVE
           5F2
4884 07D42 147          C=DAT1 A                C= ACTIVE
4885 07D45 719E         GOSUB  DT1=C-           Write and Move down 5 nibs
4886 07D49 7D8E         GOSUB  DT1=C-           Update GSBSTK, Move down 5
4887 07D4D 798E         GOSUB  DT1=C-           Update FORSTK, Move down 5
4888 07D51 145          DAT1=C A                Update MTHSTK
4889                *
4890                * ZERO CNTADR --> TMRAD3
4891                *
4892 07D54      =ZERPGM
4893 07D54 28           P=     15-((TMRAD3)-(CNTADR))/5
4894 07D56 D2           C=0    A
4895 07D58 1FE7         D1=(5) =CNTADR           First Pointer to Zero
           6F2
4896 07D5F 8E00         GOSUBL =initpt
           00
4897                *
4898                * ZERO PARAMETER COUNT
4899                *
4900 07D65 1E7B         D1=(4) =PRMPTR
           5F
4901 07D6B 14D          DAT1=C B
4902                *
4903                * Clear SUSP Annunciator
4904                *
4905 07D6E 7F1C         GOSUB  CLSUSP           Clear suspend annunciator
4906                *
4907                * Zero Alarm RAM for Timers   (PUTALM)
4908                *   Only C(0) is important
4909                * Return Set Carry
4910                *
4911 07D72 302          LC(1)  2                3 Timers
4912 07D75 D7           D=C    A                Counter
4913 07D77 AF0          A=0    W                Timer interval = 0
4914 07D7A DB    CLAP10 C=D    A                Timer # - 1
4915 07D7C 8F00         GOSBVL =PUTALM          C(S) = Timer #-1
           000
```

```
4916 07D83 A0F          D=D-1  P              Timer# counter
4917 07D86 53F          GONC   CLAP10         Clear all 3 timers
4918               *
4919               * Fast Poll to allow future statements
4920               *   to zero addresses or RAM associated with the statement
4921               *
4922 07D89 7398         GOSUB  FPoll          Fast Poll
4923 07D8D 7F           CON(2) =pZERPG        to zero addresses/RAM
4924 07D8F 03           RTNCC                 Return with carry clear
4925               *
4926 07D91 137   CLAP20 CD1EX
4927 07D94 134          DO=C                DO PTS AT CALSTK POINTER
4928               *
4929 07D97 131   CLAP30 D1=A                D1 PTS AT CALSTK
4930 07D9A 8F00         GOSBVL =MANSTK      GET TO MAINFRAME CALL STACK
           000
4931 07DA1 133          AD1EX               A= MAINFRAME CALL STACK ADDR
4932 07DA4 D2           C=0    A
4933 07DA6 31D3         LC(2)  (caACTV)-(caCURS)+1
4934 07DAA CA           A=A+C  A             A POINTS TO SAVED OFFSET OF ACTIVE
4935 07DAC 131          D1=A
4936 07DAF 184          DO=DO- 5             DO POINTS TO ACTIVE
4937 07DB2 8F00         GOSBVL =RSTOPT       RESTORE ACTIVE
           000
4938 07DB9 164          DO=DO+ 5
4939 07DBC 8F00         GOSBVL =RSTOPT       RESTORE CALSTK
           000
4940               *       D1=(5) =RAMEND
4941               *       A=DAT1 A
4942               *       ACEX   A
4943 07DC3 DE           ACEX   A
4944 07DC5 8F00         GOSBVL =C=RAME
           000
4945 07DCC 8A6          ?A#C   A             REACH DOWN TO LAST LEVEL YET ?
4946 07DCF 8C           GOYES  CLAP30        IF NOT, KEEP GOING
4947 07DD1 766F         GOSUB  CLRSTK
4948               *
4949               * RBLDCH will clear carry on exit
4950               *
4951 07DD5 8D00         GOVLNG =RBLDCH       REBUILD CHAIN HEAD POINTERS
           000
4952               *
4953 07DDC              END
```

ACTIVE	Abs	193960	#2F5A8	-	14	4883		
ALINFO	Ext			-	589			
ALMSRV	Ext			-	2374			
ASNMNT	Ext			-	2230			
=ATCHK	Abs	30679	#077D7	-	3270			
ATNFLG	Abs	193602	#2F442	-	14	2785		
AUTCLR	Ext			-	739	1138		
AUTLN2	Ext			-	1503			
AUTOCK	Ext			-	1522			
=BASCHR	Abs	30529	#07741	-	3087			
=BASCHK	Abs	30526	#0773E	-	3086	668	713	2449
BASCK3	Abs	30544	#07750	-	3091	3027		
BASICs	Abs	181	#000B5	-	13	2207		
=BLANKC	Abs	30744	#07818	-	3373	3316		
BLDDSP	Ext			-	1222			
=BLNKC+	Abs	30736	#07810	-	3371			
BSCEX+	Abs	30215	#07607	-	2495	2450	2452	2462
=BSCEX2	Abs	29754	#0743A	-	2144	904		
=BSCEXC	Abs	29751	#07437	-	2143			
=BSCEXT	Abs	30159	#075CF	-	2446	2182	2750	
BSCX10	Abs	29780	#07454	-	2168			
BSCX20	Abs	29790	#0745E	-	2180			
BSCX22	Abs	29799	#07467	-	2183	2181		
BSCX25	Abs	29821	#0747D	-	2189	2187		
BSCX30	Abs	29827	#07483	-	2195	2156	2171	
BSCX40	Abs	29855	#0749F	-	2207			
BSCX45	Abs	29887	#074BF	-	2230	2209		
BSCX50	Abs	30142	#075BE	-	2404	2307	2334	2342
BSCX60	Abs	30150	#075C6	-	2411	2253		
BSCX70	Abs	30159	#075CF	-	2447	2412		
BSCX85	Abs	30208	#07600	-	2475	2468		
BSCX95	Abs	30227	#07613	-	2498	2496		
=BSCXLP	Abs	29777	#07451	-	2167	2413		
BSERR	Ext			-	528			
C=RAME	Ext			-	4875	4944		
=CHAIN	Abs	28473	#06F39	-	486			
=CHAIN*	Abs	31793	#07C31	-	4599	4473		
=CHAIN+	Abs	31762	#07C12	-	4589	738		
=CHAIN-	Abs	31772	#07C1C	-	4591			
CHAINP	Ext			-	485			
=CHKPSF	Abs	30586	#0777A	-	3175			
CHN110	Abs	31837	#07C5D	-	4613	4610		
CHN120	Abs	31846	#07C66	-	4616	4655		
CHN130	Abs	31862	#07C76	-	4622	4620		
CHN140	Abs	31870	#07C7E	-	4626			
CHN180	Abs	31911	#07CA7	-	4650	4683	4700	
CHN190	Abs	31916	#07CAC	-	4652	4696		
CHN250	Abs	31945	#07CC9	-	4665	4641		
CHN280	Abs	31947	#07CCB	-	4666	4644		
CHN300	Abs	31927	#07CB7	-	4657	4624		
CHN510	Abs	31961	#07CD9	-	4671	4632	4635	4638
CHN520	Abs	31986	#07CF2	-	4680	4678		
CHN530	Abs	32032	#07D20	-	4698	4691		
=CK"ON"	Abs	30381	#076AD	-	2785	2331		
CKEX-0	Abs	29898	#074CA	-	2248	2300		

```
 CKEX00  Abs   29915 #074DB -  2253  2291
 CKEX01  Abs   29919 #074DF -  2254  2252
 CKEX05  Abs   29972 #07514 -  2302  2297
 CKEX06  Abs   29988 #07524 -  2307  2332
 CKEX07  Abs   29992 #07528 -  2313  2306
 CKEX08  Abs   30007 #07537 -  2331  2256
 CKEX10  Abs   30031 #0754F -  2338  2367  2369
 CKEX20  Abs   30041 #07559 -  2340  2346
 CKEX30  Abs   30125 #075AD -  2394  2381
 CKSREQ  Ext                -  2304
 CLAP10  Abs   32122 #07D7A -  4914  4917
 CLAP20  Abs   32145 #07D91 -  4926  4879
 CLAP30  Abs   32151 #07D97 -  4929  4946
 CLOSEA  Ext                -  2717
=CLPSTK  Abs   32041 #07D29 -  4875   684   733  2739
=CLRSTK  Abs   32059 #07D3B -  4883  2716  4947
=CLSUSP  Abs   31121 #07991 -  3718  1322  4905
 CNTADR  Abs  194174 #2F67E -    14   965  1343  4893  4895
 CNTCHK  Abs   29177 #071F9 -   963   731   847
=CNTCK2  Abs   29182 #071FE -   965  1149  4118
=CNTCUR  Abs   29576 #07388 -  1360  2476  4222
 COLLAP  Ext                -   833
=COMPL#  Abs   30832 #07870 -  3506
=CONT    Abs   28506 #06F5A -   508
=CONTK   Abs   28480 #06F40 -   491
 CONTP   Ext                -   507
 COPYu   Ext                -   628
=CPL#10  Abs   30855 #07887 -  3520  1361  3510  4340
=CPL#15  Abs   30858 #0788A -  3526  1197
 CPL#25  Abs   30889 #078A9 -  3538  3551
 CPL#30  Abs   30905 #078B9 -  3544  3552
 CPL#60  Abs   30930 #078D2 -  3553  3540
 CPL#70  Abs   30932 #078D4 -  3554  3516
 CRLFOF  Ext                -  1257
 CURO20  Ext                -  1548
 CURDVC  Ext                -   610
 CURRLO  Ext                -  1544
 CURRST  Abs  193885 #2F55D -    14  2903  4589
=CURSNP  Abs   29714 #07412 -  1522
 CrLfof  Abs   29469 #0731D -  1257   498  1137  1249
 D0=PCA  Ext                -   770  1012
 D1=TFB  Abs   29475 #07323 -  1262  1176  1206  1253
 DATPTR  Abs  194194 #2F692 -    14  4201
 DCPLIN  Ext                -  1487
 DEBNCE  Ext                -  1234
=DEFEND  Abs   30247 #07627 -  2663  2691
 DLFIBA  Ext                -   737  1152
 DSPCNO  Ext                -  1219
 DT1=C-  Abs   31706 #07BDA -  4467  4885  4886  4887
 EDIT20  Ext                -   694
=END     Abs   30271 #0763F -  2677
=END10   Abs   30283 #0764B -  2688  1252  2188  2732
=END20   Abs   30308 #07664 -  2715
 END25   Abs   30308 #07664 -  2716
 END30   Abs   30312 #07668 -  2717  2740
```

```
 END40   Abs   30339 #07683 -  2725  2723
 ENDAL1  Abs   30365 #0769D -  2739  2680
=ENDALL  Abs   30362 #0769A -  2738
=ENDBIN  Abs   30283 #0764B -  2687
 ENDDC   Ext                -  2667
 ENDDEF  Ext                -  2663
 ENDP    Ext                -  2668
 ENDSB-  Ext                -  2665
 EOLSCN  Ext                -  2469
=ERRRTN  Abs   29933 #074ED -  2278
 ERRSUB  Abs  194179 #2F683 -    14  4107  4267
 EXCADR  Ext                -  2214
=EXITRN  Abs   30375 #076A7 -  2749  2719
 Except  Abs      12 #0000C -    13  2296  2305  2313
 FCHLB2  Abs   30768 #07830 -  3424  3443
 FCHLB4  Abs   30783 #0783F -  3428  3426
 FCHLB6  Abs   30827 #0786B -  3447  3436
=FCHLBL  Abs   30764 #0782C -  3423  1516
 FDLB10  Abs   30617 #07799 -  3248  3263
 FDLB20  Abs   30635 #077AB -  3254  3247  3268
 FDLB30  Abs   30661 #077C5 -  3264  3253
=FETCH   Abs   29596 #0739C -  1450
 FETCHP  Ext                -  1449
 FILXQT  Abs   29499 #0733B -  1311  4161
 FILXQ^  Ext                -  1311
 FINDF   Ext                -   649
=FINDLB  Abs   30598 #07786 -  3243   831  4169
 FINDLR  Ext                -   789  1481
 FLUSHA  Ext                -  2497
 FNDDO+  Ext                -  4629
 FORSTK  Abs  193950 #2F59E -    14  2279
 FPOLL   Ext                -  2502
=FPoll   Abs   30240 #07620 -  2502  2146  2314  2498  4922
 FSPCER  Ext                -   556
 FSPECj  Ext                -   526
 FTCO10  Abs   29618 #073B2 -  1460  1453
 FTCO20  Abs   29631 #073BF -  1470
 FTCO25  Abs   29655 #073D7 -  1481  1545
 FTCO30  Abs   29665 #073E1 -  1486  1518
 FTCO35  Abs   29675 #073EB -  1500  1547
 FTCO40  Abs   29679 #073EF -  1503  1482
 FTCO50  Abs   29689 #073F9 -  1511  1476
 FTCO60  Abs   29727 #0741F -  1544  1514  1523
 FTCER2  Abs   29685 #073F5 -  1507  1512  1517  4279
 FTCHKY  Ext                -  1455
 GETPRO  Ext                -  3016  3170
=GETPRe  Abs   30572 #0776C -  3169  1139  1466  3175
=GETPeF  Abs   30493 #0771D -  3016
=GETST*  Abs   30486 #07716 -  3013
=GETST-  Abs   30504 #07728 -  3020  3528  3782
=GETST1  Abs   30508 #0772C -  3021  3014
=GETSTC  Abs   30502 #07726 -  3019   748   849  2718  3101  3423  4421
=GETSTe  Abs   30559 #0775F -  3101  3169  4428
 GETSe1  Abs   30563 #07763 -  3102  3171
 GOSBDC  Ext                -  4046
```

```
=GOSUB   Abs   31209 #079E9 -  4049
 GOSUBP  Ext                 4047
=GOTO    Abs   31226 #079FA -  4055
=GOTO+   Abs   31238 #07A06 -  4062
 GOTO-   Abs   31229 #079FD -  4056  4050
 GOTODC  Ext                 4052
 GOTOP   Ext                 4053
 GTMRA+  Ext                 2364
 GTO001  Abs   31232 #07A00 -  4057  4037
 GTO005  Abs   31248 #07A10 -  4072
 GTO010  Abs   31262 #07A1E -  4084  4073
 GTO020  Abs   31285 #07A35 -  4096  4090
 GTO030  Abs   31314 #07A52 -  4115  4106
 GTO040  Abs   31348 #07A74 -  4129  4116
 GTO050  Abs   31359 #07A7F -  4136  4098  4109  4124  4130
 GTO060  Abs   31365 #07A85 -  4143  4086
 GTO110  Abs   31420 #07ABC -  4182  4151
 GTO120  Abs   31433 #07AC9 -  4197  4172
 GTO130  Abs   31451 #07ADB -  4218  4198
 GTO140  Abs   31469 #07AED -  4229  4219  4221
 GTO150  Abs   31492 #07B04 -  4239  4231
 GTOLE0  Abs   31500 #07B0C -  4258  4162  4170  4183
 GTOLE1  Abs   31520 #07B20 -  4264  4259
 GTOLE2  Abs   31539 #07B33 -  4269  4265
 GTOLE3  Abs   31559 #07B47 -  4278  4270
 GTOLE4  Abs   31565 #07B4D -  4279
 ISRAM?  Ext                 3705  4607
 KEY?    Ext                 1231
=LBLNAM  Abs   30695 #077E7 -  3316  3265  3433
 LBLRCK  Abs   29487 #0732F -  1307   826  1511
 LBNM10  Abs   30705 #077F1 -  3319  3324
 LBNM20  Abs   30721 #07801 -  3325  3320
 LBNM30  Abs   30723 #07803 -  3326  3322
 LDCSET  Ext                 1198  3371
 LDSST1  Ext                 1188
 LDSST2  Ext                 1209
=LET     Abs   29887 #074BF -  2229
 LETDC   Ext                 2227
 LETP    Ext                 2228
 LIN#DC  Ext                 1199
 LNSKP-  Ext                 4091
 MAINLP  Ext                 1155
 MANSTK  Ext                 4930
 MFERR   Ext                 3103
 MLPEXT  Abs   29268 #07254 -  1155   840  1524  2501
 Mferr   Abs   30566 #07766 -  3103  3017  3178
=NOPRGM  Abs   30427 #076DB -  2831
 NOSCRL  Ext                 1220
 NOTBAS  Abs   30548 #07754 -  3093  3089
=NULLP   Abs   31129 #07999 -  3782   838  1153  1546
 NXTST1  Ext                 2726
 NXTSTM  Ext                 2828
 NoCont  Abs      14 #0000E -    13  1148  2143  2411  2500  2724  2789  2827
=Nxtstm  Abs   30421 #076D5 -  2828   801  2231  4223
 OFFTMR  Ext                 4273
```

```
 ONTIMR  Ext                  -  2398
 OUT2TC  Ext                  -  1201
=PAUSE   Abs   30418 #076D2  -  2827
 PCADDR  Abs  194169 #2F679  -    14  2195  4344
 PFDL10  Abs   31003 #0791B  -  3668  3650  3652  3655
 PFDL50  Abs   31028 #07934  -  3679  3693
 PFDL60  Abs   31053 #0794D  -  3688  3696
 PFDL65  Abs   31055 #0794F  -  3689  3674
 PFDL70  Abs   31078 #07966  -  3697  3682
 PFDL75  Abs   31119 #0798F  -  3714  3709
=PFINDL  Abs   30943 #078DF  -  3643
=PFNDL*  Abs   31026 #07932  -  3678
=PFNDZL  Abs   30946 #078E2  -  3644  4182
 PNDALM  Abs  194401 #2F761  -    14  2248  2338  2372
 POPGSB  Ext                  -  4260
 Poll1j  Ext                  -  2834
 PRGFMF  Ext                  -   611
 PRGMEN  Abs  193895 #2F567  -    14  1161  2183  3668
 PRGMST  Abs  193890 #2F562  -    14  3243  4461
 PRGROM  Ext                  -  4611
 PRMPTR  Abs  193975 #2F5B7  -    14  2688  4900
=PRSC00  Abs   31635 #07B93  -  4429  4422
 PRSC10  Abs   31685 #07BC5  -  4459  4437
=PRSC20  Abs   31691 #07BCB  -  4461  4455
 PRSC50  Abs   31714 #07BE2  -  4473  4435
=PRSC60  Abs   31700 #07BD4  -  4465
=PRSCKB  Abs   31624 #07B88  -  4425  4071
=PRSC0-  Abs   31616 #07B80  -  4421   734
=PRSCOP  Abs   31631 #07B8F  -  4428  1151  4474
 PSHGSB  Ext                  -  4136
 PSHUPD  Ext                  -  4120
 PUTALM  Ext                  -  4915
 PgmRun  Abs      13 #0000D  -    13   613   873  1323  2155  2180  2333  2451
                              2722  2748  2831  3651  4115  4220  4414
 Poll    Abs   30440 #076E8  -  2834   673   893
 RBLDCH  Ext                  -  4951
=RDCHD+  Abs   30446 #076EE  -  2902   712  2448
=RDCHDR  Abs   30448 #076F0  -  2903  3020
 RDHD10  Abs   30478 #0770E  -  2912  2909
 RDHDR0  Abs   30458 #076FA  -  2905  3013
=RDHDR1  Abs   30461 #076FD  -  2906   667
 RDINFS  Ext                  -   647
 RESTDC  Ext                  -  4020
=RESTOR  Abs   31162 #079BA  -  4022
 RESTR#  Ext                  -  4030
 RESTR1  Abs   31181 #079CD  -  4034  4029
 RESTRP  Ext                  -  4021
 RESTRX  Abs   31438 #07ACE  -  4201  4044
 RSTOPT  Ext                  -  4937  4939
 RTNCC   Ext                  -  2731  2826
=RUN     Abs   28526 #06F6E  -   517
 RUN005  Abs   28490 #06F4A  -   497   492
 RUN010  Abs   28512 #06F60  -   510   521
 RUN015  Abs   28526 #06F6E  -   518   487
 RUN017  Abs   28554 #06F8A  -   538   563
```

```
 RUN018  Abs   28559 #06F8F -   540   581
 RUN020  Abs   28566 #06F96 -   549   527
 RUN022  Abs   28589 #06FAD -   557   553   555
 RUN025  Abs   28611 #06FC3 -   572
 RUN034  Abs   28642 #06FE2 -   588   539
 RUN035  Abs   28645 #06FE5 -   589   574   579   583
 RUN040  Abs   28677 #07005 -   606
 RUN045  Abs   28708 #07024 -   623   607
 RUN047  Abs   28713 #07029 -   625   654
 RUN050  Abs   28759 #07057 -   647   624
 RUN053  Abs   28768 #07060 -   649   541
 RUN055  Abs   28787 #07073 -   666   636   652
 RUN057  Abs   28801 #07081 -   670   714
 RUN058  Abs   28847 #070AF -   691   672
 RUN060  Abs   28850 #070B2 -   692   669
 RUN065  Abs   28865 #070C1 -   699   510
 RUN070  Abs   28869 #070C5 -   711   499
 RUN075  Abs   28883 #070D3 -   731   695
 RUN076  Abs   28939 #0710B -   756   741
 RUN077  Abs   28949 #07115 -   760   757
 RUN080  Abs   28953 #07119 -   768   759
 RUN085  Abs   28995 #07143 -   780   777
 RUN090  Abs   29031 #07167 -   806   790   800
 RUN100  Abs   29041 #07171 -   813
 RUN110  Abs   29058 #07182 -   826   782
 RUN120  Abs   29089 #071A1 -   838   760   829
 RUN125  Abs   29100 #071AC -   847   839
 RUN140  Abs   29111 #071B7 -   859   815   834
 RUN145  Abs   29116 #071BC -   861   848
 RUN150  Abs   29132 #071CC -   873   860
 RUN155  Abs   29141 #071D5 -   883   874  1256
 RUN160  Abs   29144 #071D8 -   884   749
 RUN165  Abs   29166 #071EE -   897   895
 RUN170  Abs   29173 #071F5 -   904   892
 RUN773  Abs   28914 #070F2 -   738   736
 RUN775  Abs   28920 #070F8 -   739   732
 RUNDC   Ext                -   484   515
=RUNER1  Abs   29051 #07178 -   819   808   827   832  1507
 RUNER5  Abs   28651 #06FEB -   590   612
 RUNER8  Abs   28828 #0709C -   681   633   650   675
 RUNER9  Abs   28843 #070AB -   686   682
 RUNERR  Abs   28548 #06F84 -   528   590   686   896
 RUNEXT  Abs   29027 #07163 -   801   798
=RUNK    Abs   28487 #06F47 -   496
 RUNP    Ext                -   516
=RUNRT1  Abs   29927 #074E7 -  2276   875  4240
=RUNRTN  Abs   29930 #074EA -  2277
 RUNSST  Abs   30372 #076A4 -  2748  2721
=RUSUS?  Abs   31601 #07B71 -  4414  4425
 Rtncc*  Abs   30693 #077E5 -  3275  3273
 Rtncc+  Abs   31150 #079AE -  3789  3687  3700
 Rtncc-  Abs   30941 #078DD -  3557  3663
 S-R0-0  Abs  194673 #2F871 -    14  4143  4271
 S-R0-1  Abs  194678 #2F876 -    14   768  1014
 SAVEL#  Ext                - 1362
```

```
 SAVPCr  Abs   29201 #07211 -  1011   558   699
 SCOPCK  Ext                -  2791
=SCOPEN  Abs   31722 #07BEA -  4512  4450
=SETSU1  Abs   29549 #0736D -  1346
 SETSU2  Abs   29568 #07380 -  1352  1350
=SETSUS  Abs   29536 #07360 -  1343  1360
=SFGPGM  Abs   29519 #0734F -  1322   884  4239
 SFLAGs  Abs   29530 #0735A -  1325  1353
=SFlag?  Abs   31610 #07B7A -  4418
 SNcrlf  Ext                -  4278
 SPEN10  Abs   31739 #07BFB -  4522  4515
 SPEN20  Abs   31757 #07C0D -  4536  4526
=SST     Abs   29224 #07228 -  1137
 SST010  Abs   29238 #07236 -  1148
 SST020  Abs   29274 #0725A -  1161  1150
 SST025  Abs   29299 #07273 -  1175  1154
 SST030  Abs   29302 #07276 -  1176  1166
 SST040  Abs   29336 #07298 -  1196  1186
 SST050  Abs   29383 #072C7 -  1219  1189
 SST055  Abs   29411 #072E3 -  1231  1168
 SST060  Abs   29420 #072EC -  1233  1239
 SST070  Abs   29443 #07303 -  1249  1232
 SST080  Abs   29456 #07310 -  1253  1251
 STMBCL  Ext                -   863
 STMIDO  Abs  194705 #2F891 -    14   626   629
 STMINF  Ext                -   819
=STOP    Abs   30358 #07696 -  2732
 STOPDC  Ext                -  2730  2825
 SUBCHK  Ext                -  2692
 SUBEND  Abs   30254 #0762E -  2665  2693
 SVINF+  Ext                -   592
 SVINF0  Ext                -   596
=Scopck  Abs   30402 #076C2 -  2791   807  1348  2368  3659
=SflagC  Abs   30434 #076E2 -  2833  3719
=SflgCp  Abs   30430 #076DE -  2832  2447
 TKSCN7  Ext                -  3425
 TMRAD3  Abs  194209 #2F6A1 -    14  4893
 TRFCK-  Ext                -  4281
 TRFMBF  Abs  194757 #2F8C5 -    14  1262
 TRFROM  Ext                -  4074
 TRT0+   Ext                -  4232
=TrfCk-  Abs   31569 #07B51 -  4281  2380  4072  4230
=UPDPC   Abs   31576 #07B58 -  4340  2386
=UPDPCC  Abs   31589 #07B65 -  4344
 USRSTA  Ext                -  2316
=ZERPGM  Abs   32084 #07D54 -  4892
 caACTV  Abs      66 #00042 -    13  4933
 caCURS  Abs       6 #00006 -    13  4933
 cursfl  Ext                -  1221
 dCARD   Abs       7 #00007 -    13   550
 dPORT   Abs       1 #00001 -    13   576
=dcplin  Abs   29668 #073E4 -  1487
 eFTYPE  Ext                -   676  3093
 fBASIC  Abs   57876 #0E214 -    13  3087
 fBIN    Abs   57860 #0E204 -    13   670
```

```
 flPRGM  Abs     -62 #FFFC2 -    13  1324  2832
 flSUSP  Abs     -63 #FFFC1 -    13  1352  3718  4417
 initpt  Ext               -  4896
 lEOL    Abs       2 #00002 -    13  3025
 nxtstm  Abs   31465 #07AE9 -  4223  4203
 oBSsod  Abs      17 #00011 -    13  3025  3786
 oFLENh  Abs      32 #00020 -    13   692  2912  4592
 oFNAMh  Abs       0 #00000 -    13   693
 oFTYPh  Abs      16 #00010 -    13   692   693  2906  2912
 pBSCen  Abs     245 #000F5 -    13  2147
 pBSCex  Abs     246 #000F6 -    13  2499
 pExcpt  Abs     248 #000F8 -    13  2315
 pRUNft  Abs      48 #00030 -    13   674
 pRUNnB  Abs      49 #00031 -    13   894
 pZERPG  Abs     247 #000F7 -    13  4923
 sCHAIN  Abs      11 #0000B -    13   486   538   580   606   651   681   735
                                758
 sCONT   Abs      10 #0000A -    13   491   496   508   859   963
 sCONTK  Abs       9 #00009 -    13   497   509   756
 sDEST   Abs       3 #00003 -    13   559   595
 sENDx   Abs       1 #00001 -    13  2276  2461  2725
 sERROR  Abs       0 #00000 -    13  2277  2290  2406  2495
 sEXTGS  Abs       5 #00005 -    13  4058  4089
 sGOSUB  Abs       3 #00003 -    13  4049  4055  4085  4218  4258
 sMAINc  Abs       5 #00005 -    13   560   588   623
 sONERR  Abs       4 #00004 -    13  2397  4057  4105  4264
 sONTMR  Abs       6 #00006 -    13  2396  4129  4269
 sRESTR  Abs      10 #0000A -    13  4034  4056  4197
 sRUNBn  Abs       4 #00004 -    13   666   691   711   740   883   891
 sSST    Abs       2 #00002 -    13  1167  1223  1250  2720  2738
 sSTdc   Abs       1 #00001 -    13  1178
 sXWORD  Abs       9 #00009 -    13  3643  3649  4097
=savel#  Abs   29584 #07390 -  1362
 sflag?  Ext               -  4418
 sflagc  Ext               -  2833
 sflags  Ext               -  1325
 tALL    Ext               -  2678
 tCOMMA  Ext               -   775
 tDEF    Ext               -  4634
 tELSE   Ext               -  2465
 tENDDF  Ext               -  4637
 tENDSB  Ext               -  4643
 tKEY    Ext               -  1450
 tLBLRF  Ext               -  1308
 tLBLST  Ext               -  3249  3424  4631
 tLINE#  Ext               -   780  1474  4149
 tRFILE  Ext               -   519
 tSUB    Ext               -  4513  4640
```

Input Parameters

  Source file name is JP&SYS::MS

  Listing file name is JP/SYS:TI:ML::-1

  Object file name is JP%SYS:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *        J  PPPP    &     EEEEE  X   X   CCC
 2            *        J  P   P  & &    E      X   X  C   C
 3            *        J  P   P  & &    E       X X   C
 4            *        J  PPPP    &     EEEE     X    C
 5            *        J  P      & & &  E       X X   C
 6            *    J   J  P      &  &   E      X   X  C   C
 7            *     JJJ   P       && &  EEEEE  X   X   CCC
 8            *
 9                      TITLE  Program Execution Routines <831212.1515>
10 07DDC               ABS    #07DDC
11                      RDSYMB TI%EQU::MS
12                      RDSYMB SB%RAM::MS
13           ********************************************************************
14           ********************************************************************
15           **
16           ** Name:    USER    -  Statement Execute
17           ** Name:    LC      -  Statement Execute
18           **
19           ** Category:   STEXEC
20           **
21           ** Purpose:
22           **      Execute USER | LC statement
23           **
24           ** Entry:
25           **      USER: DO past USER token
26           **      LC:   DO past LC token
27           **
28           ** Exit:
29           **      Through NXTSTM
30           **
31           ** Calls:      SFLAGT,SFLAG*
32           **
33           ** Uses.......
34           **  Exclusive: A(B),B(A),C(A),DO
35           **  Inclusive: A(B),B(A),C(A),DO,C(S),D(A),P
36           **             ANNAD1-4,SYSFLG
37           **
38           ** Stl lvls:   4
39           **
40           ** NOTE:
41           **      tOFF must IMMEDIATELY follow tON
42           **      There is a HARD WIRED (B=B+1) in this code
43           **
44           ** Detail:
45           **      USER [ ( ON | OFF ) ]
46           **      LC [ ( ON | OFF ) ]
47           **
48           ** History:
49           **
50           **    Date     Programmer            Modification
51           **  --------   ----------   ---------------------------------
52           **  07/04/82   JP           Modified documentation
53           **  11/16/82   JP           Interfaced to SFLAG*
54           **
55           **
```

```
56            ********************************************************************
57            ********************************************************************
58 07DDC 0000         REL(5) =FLIPDC
         0
59 07DE1 0000         REL(5) =FLIPP
         0
60 07DE6      =LC
61 07DE6 311F =FLIP   LC(2)  =flLC          LC flag number
62 07DEA 6110         GOTO   USER10
63 07DEE 0000         REL(5) =USERDC
         0
64 07DF3 0000         REL(5) =USERP
         0
65 07DF8 317F =USER   LC(2)  =flUSER        USER flag number
66 07DFC D5    USER10 B=C    A              Save flag number
67 07DFE 3100         LC(2)  =tON
68 07E02 DD           BCEX   A              B <-- Token, C <-- Flag number
69 07E04 14A          A=DATO B              Read next token
70 07E07 960          ?A=B   B              ON token ?
71 07E0A 41           GOYES  USER20         Yes; turn on flag/annunciator
72            *
73            * NOTE ***************************************************
74            *   tOFF must IMMEDIATELY follow tON
75            *
76 07E0C E5           B=B+1  A              OFF token
77 07E0E 960          ?A=B   B              OFF token ?
78 07E11 F0           GOYES  USER30         Yes; turn off flag/aannunciator
79            *
80            * Toggle flag/annunciator
81            *
82 07E13 8F00         GOSBVL =SFLAGT        Toggle flag/annunciator
         000
83 07E1A 6444  USREXT GOTO   Nxtstm         Return to Run loop
84            *
85            *  ON - Set flag/annunciator
86            *
87 07E1E 21    USER20 P=     1              "Set" flag indicator
88            *
89            * OFF - Clear flag/annunciator
90            *       P=0 ---> Clear flag
91            *       P=1 ---> Set   flag
92            *
93 07E20 8F00  USER30 GOSBVL =SFLAG*        Set/Clear flag
         000
94 07E27 52F          GONC   USREXT         B.E.T.
```

```
                STITLE ON Statement Execution
*****************************************************************
*****************************************************************
**
** Name:    ON      -  Statement Execute
**
** Category:   STEXEC
**
** Purpose:
**      ON Statement Execute
**
** Entry:
**      DO past ON token
**
** Exit:
**      If ON ERROR | TIMER ---> Return thru NXTSTM
**      If ON GOSUB ---> Execute thru ONGSB   (EXCERR,GOSUB)
**      If ON GOTO | RESTORE ---> Execute thru ONGTRS
**
**      Error Exits - "Invalid Arg"
**         Negative Expression (from RNDEXP/FLTDH)
**         Bad timer#          (from GTMR#)
**         ON <exp>=0 or Out of range
**
**         ARGSTA Error Exits: for Complex,Array,Proj Inf
**
** Calls:      RNDEXP, ACTTMR, EXPEX-, GTMR#, GTMRAD, SEC2TK,
**             ARDSTA, D1=SR0, TMIADR, EXPSKP
**
** Uses.......
**  Exclusive: A-D,D1,D0,S3,S5,S10,ERRADR,TMRAD1-3
**             R0,R1,STMTR0 (5 nibs),TMRIN1-TMRIN3
**
** Stk lvls:   <=7  (Statement Execute routine)
**
**  S3= GOSUB
**  S5= External entry flag for GOSUB|GOTO|RESTORE execute
**  S10= ON RESTORE
**  R0 = Timer Interval
**  R1 = Timer #
**
** Detail:
**      If ON ERROR
**         Zero ERRSUB address
**         Save address of GOTO|GOSUB in stmt into ERRADR
**         go Skip to end of stmt & return to Run Loop (NXTSTM)
**      If ON TIMER
**         Move past TIMER token
**         Get Timer#                          (GTMR#)
**         Save Timer#                         (STMTR0)
**         Skip comma
**         Evaluate <#secs>                    (EXPEX-)
**         Standard arguement, Error if bad    (ARGSTA)
**         If negative                         (A(S)#0)
**            Interval <-- minimum             (1)==1/32
```

```
150             **          goto 2;
151             **       Convert Seconds to Ticks (1/512) (SEC2TK)
152             **       Shift for 1/32 accuracy
153             **       If interval < 1
154             **          Interval <-- minimum          (1)
155             **          goto 2;
156             **       If interval > Maximum          (FFFFFFFF)
157             **          Interval <-- Maximum
158             **    2:    Save interval                  (R0)
159             **       Get ON TIMER address pointer     (GTMRAD)
160             **       Save TImer Address               (R1)
161             **       Set ON TIMER address
162             **       Compute offset to Timer Int field (TMIADR)
163             **       Shift Timer Interval for 1/512 accuracy
164             **       Activate Timer                   (ACTTMR)
165             **       go to Next statment              (NXTSTM)
166             **     If ON-GOTO | GOSUB | RESTORE
167             **       Evaluate Branch Number and round (RNDEXP)
168             **       If Branch Number = 0   (Decrement & Carry)
169             **          Error Exit ---> "Invalid Arg"
170             **       Set B = Branch number
171             **       If GOSUB ---> Set sGOSUB  (S3)
172             **       If RESTORE -> Set sRESTR  (S10)
173             **  1:   Skip statement type token or comma
174             **       Decrement Branch number
175             **       If Branch number # 0   (Carry Clear)
176             **          Read next token
177             **          If token = line#  (0F)
178             **             Skip line# token & line#
179             **             Read next token
180             **          If token = label  (F6)
181             **             If tLITRL label
182             **               Skip until comma | EOL | @ | ! (A(1)=F)
183             **             else
184             **               Skip label expression          (EXPSKP)
185             **          If token = comma  (F1)
186             **             goto 1;
187             **          else ---> Error Exit   (Branch # too large)
188             **       If Branch number = 0   (Carry set)
189             **          Clr External entry flag for GOTO|GOSUB exec (sEXTGS
190             **          go Execute GOTO | RESTORE | GOSUB  (ONGTGB)
191             **
192             ** History:
193             **
194             **    Date     Programmer            Modification
195             **  --------   ----------   ----------------------------------
196             **  07/04/82   JP           Modified documentation
197             **  09/04/82   JP           Interfaced to ARGSTD
198             **  10/14/82   JP           Interfaced to ARGSTA
199             **  11/29/82   JP           Change ON...GOSUB interface
200             **  03/08/83   JP           Clear sEXTGS before ONGTGB jump
201             **  05/11/83   JP           RNDEXP errors if Branch#=0
202             **  06/30/83   JP           Call EXPSKP is "Label" expression
203             **
204             ****************************************************************
```

```
205             ****************************************************************
206 07E2A 0000          REL(5) =ONDC            ON statement Decompile
          0
207 07E2F 0000          REL(5) =ONP             ON statement Parse
          0
208 07E34 14A  =ON      A=DAT0 B                Read type of ON statement
209 07E37 3100          LC(2)  =tERROR          ERROR token
210 07E3B 966           ?A#C   B                Not ON ERROR ?
211 07E3E E1            GOYES  ON010
212             *
213             * ON ERROR Execute
214             *    Zero ERRSUB address - to negate prior ON ERROR - GOSUB
215             *
216 07E40 1F38          D1=(5) =ERRSUB          Return address for ON ERROR - GOSUB
          6F2
217 07E47 D2            C=0    A                Zero it
218 07E49 145           DAT1=C A
219 07E4C 174           D1=D1+ (ERRADR)-(ERRSUB)  Move to ERRADR
220 07E4F 161           D0=D0+ 2                Move past ERROR token
221 07E52 136           CD0EX                   Save addr to GOTO|GOSUB in ON ERROR
222 07E55 145           DAT1=C A
223 07E58 6604  ONEXIT  GOTO   Nxtstm           Skip to end of stmt, Rtn to Run Loo
224             *
225             * ON TIMER execute
226             *
227 07E5C B06   ON010   C=C+1  P                TIMER token
228 07E5F 962           ?A=C   B                ON TIMER ?
229 07E62 60            GOYES  ON015
230 07E64 6080          GOTO   ON300
231             *
232             * Evaluate Timer#, Round, Check if within bounds
233             * Save it
234             *
235 07E68 7E52  ON015   GOSUB  GTMR#
236 07E6C 7D01          GOSUB  D1=sR0           Set D1 @ STMTR0
237 07E70 141           DAT1=A A                Save Timer# in STMTR0
238             *
239             * Skip comma
240             * Evaluate <interval> from Collasped Expression stack
241             * Standard argument
242             *
243 07E73 161           D0=D0+ 2                Skip comma
244 07E76 8E00          GOSUBL =EXPEX-          Evaluate <interval>
          00
245 07E7C 8E00          GOSUBL =ARGSTA          Standardize arguement/ Error if bad
          00
246 07E82 04            SETHEX                  Restore Hexadecimal mode
247             *
248             * If negative interval
249             *    Interval <-- Minimum
250             *
251 07E84 948           ?A=0   S                Not negative number ?
252 07E87 A0            GOYES  ON035
253 07E89 AF2           C=0    W
254 07E8C E6    ON030   C=C+1  A                Min value = 1  (1/32)
```

```
255 07E8E 512          GONC   ON040          B.E.T.  (Carry clear)
256             *
257             * Convert seconds to 1/512 seconds
258             * Save accuracy to 1/32
259             *
260 07E91 8F00  ON035  GOSBVL =SEC2TK        Convert to 1/512
          000
261 07E98 BF6          CSR    W             1/32 accuracy
262 07E9B 97A          ?C=0   W             < Minimum ?
263 07E9E EE           GOYES  ON030
264             *
265             * If interval > Maximum  (FFFFFFFF)
266             *    Interval <-- Maximum
267             *
268 07EA0 27           P=     7             Max interval = 8 nibs
269 07EA2 A90          A=0    WP
270 07EA5 A1C          A=A-1  WP            Max value = FFFFFFFF
271 07EA8 99E          ?C<=A  WP            Interval < Maximum ?
272 07EAB 50           GOYES  ON040
273 07EAD AF6          C=A    W             Interval <-- Maximum
274             *
275             * Save Interval
276             *
277 07EB0 20    ON040  P=     0             Reset P
278 07EB2 108          RO=C                 Save interval
279             *
280             * Get Timer address for Timer#, Save to R1
281             * Set TIMER address = D0
282             *
283 07EB5 74C0         GOSUB  D1=sRO        Retrieve Timer#
284 07EB9 147          C=DAT1 A
285 07EBC 109          R1=C                 Save Timer#
286 07EBF 7C32         GOSUB  GTMRAD
287 07EC3 132          ADOEX                Pointer within ON TIMER statement
288 07EC6 141          DAT1=A A             Set TMRADR for Timer#
289             *
290             * Compute Timer Interval RAM location for Timer#
291             *   C = Timer# * 4
292             *
293 07EC9 7C12         GOSUB  TMIADR        Compute Timer Interval RAM loc
294 07ECD 110          A=RO                 Timer Interval
295 07ED0 1597         DAT1=A 8             Interval = 8 nibs
296             *
297             * Shift Interval to 1/512 accuracy
298             * Activate Timer
299             * goto Next statement
300             *
301 07ED4 BF0          ASL    W             Interval to 1/512 accuracy
302 07ED7 119          C=R1                 Timer#
303 07EDA 8F00         GOSBVL =ACTTMR       Activate Timer
          000
304 07EE1 667F         GOTO   ONEXIT        Return to Run Loop
305             *
306             * ON <exp> GOTO | GOSUB | RESTORE execute
307             *    If GOSUB
```

```
                        *       To save Branch number --- use D(A)
                        *
07EE5 843    ON300      ST=0    sGOSUB          Clear flags
07EE8 84A               ST=0    sRESTR
07EEB 71B1              GOSUB   RNDEXP          Evaluate Branch #
07EEF CC                A=A-1   A               Branch # = 0 ?
07EF1 D6                C=A     A
07EF3 D7                D=C     A               Move branch# to D(A)
07EF5 14A               A=DAT0  B               Read branch type
07EF8 300               LC(1)   =tGOTO          GOTO low nibble
07EFB 902               ?A=C    P               GOTO ?
07EFE 31                GOYES   ON500
07F00 A0E               C=C-1   P               GOSUB low nibble
07F03 902               ?A=C    P               GOSUB ?
07F06 80                GOYES   ON400
07F08 85A               ST=1    sRESTR          RESTORE flag
07F0B 550               GONC    ON500           B.E.T.
07F0E 853    ON400      ST=1    sGOSUB          GOSUB flag
                        *
                        * Loop until Branch label or line# found | End of statement
                        *
07F11 161    ON500      DO=DO+ 2                Skip branch type | comma
07F14 CF                D=D-1   A               Decrement Branch #
07F16 4F5               GOC     ON999           Branch # found
                        *
                        * Skip line# branch
                        *
07F19 14A               A=DAT0  B               Read line#|label token
07F1C 3100              LC(2)   =tLINE#         Line# token
07F20 966               ?A#C    B
07F23 B0                GOYES   ON700
07F25 16A               DO=DO+ 6+5              Skip token & line#
07F28 14A               A=DAT0  B               Read next char
07F2B 5E3               GONC    ON900           B.E.T.
                        *
                        * Skip Label branch
                        *
                        * NOTE:  Assumes Token adjacency of tLINE# and tLBLRF
                        *        tLINE# = 0F,   tLBLRF = 0E
                        *
                        * If "literal" label
                        *    Skip over ASCII until "high" bit set (EOL,",",`,!)
                        * else
                        *    Call Expression Skip
                        *
07F2E CE     ON700      C=C-1   A               Label token
07F30 966               ?A#C    B
07F33 73                GOYES   ON900
07F35 161               DO=DO+ 2                Skip Label token
07F38 14A               A=DAT0  B               Read token
07F3B 3100              LC(2)   =tLITRL
07F3F 966               ?A#C    B               Label "expression"?
07F42 51                GOYES   ON850
07F44 21                P=      1
07F46 161    ON800      DO=DO+ 2                Skip til EOL|,|@|!
```

```
363 07F49 14A          A=DAT0 B
364 07F4C B04          A=A+1  P              High nib=F if done
365 07F4F 56F          GONC   ON800
366 07F52 20           P=     0
367 07F54 451          GOC    ON900          B.E.T
368               *
369               * Skip Label expression
370               *   Entry:  D1 @ expression start
371               *   Exit:   D1 past expression
372               *           A holds  NEXT token after expression
373               *
374 07F57 136   ON850  CDOEX
375 07F5A 135          D1=C                  D1 @ Start of expression
376 07F5D 8F00         GOSBVL =EXPSKP        Skip expression
          000
377 07F64 137          CD1EX
378 07F67 134          D0=C                  D0 past expression
379               *
380               * End of label | line#
381               *   Check for comma  (F1)
382               *   If not comma ---> Error  (End of statement w/o branch found)
383               *
384 07F6A A0C   ON900  A=A-1  P              Low nib of comma = 1
385 07F6D 908          ?A=0   P              Comma ?
386 07F70 1A           GOYES  ON500
387               *
388               * Error Exit - "Invalid Arg"
389               *    Bad Timer# or <exp> or Branch #
390               *
391 07F72 6741  ONER1  GOTO   EXPER2         "Invalid Arg"
392               *
393               * Branch # = 0, @ Branch address
394               *
395 07F76 845   ON999  ST=0   sEXTGS         Clear External Stmt flag
396 07F79 62F0         GOTO   ONGTGB         Excute GOTO/GOSUB
397               *
398               *
399 07F7D 1F17  D1=sR0 D1=(5) =STMTR0
          8F2
400 07F84 01           RTN
```

```
401                 STITLE OFF Statement Execution
402             ********************************************************************
403             ********************************************************************
404             **
405             ** Name:     OFF      -  Statement Execute
406             ** Name:     BYE      -  Statement Execute
407             ** Name:     OFFTMR   -  Statement Execute
408             **
409             ** Category:   STEXEC
410             **
411             ** Purpose:
412             **      Execute OFF | BYE statement
413             **      OFF Timer if branch not found in ON TIMER execute
414             **
415             ** Entry:
416             **      OFF:      DO past OFF token
417             **      BYE:      DO past BYE token
418             **      OFFTMR:   A(0) = Timer#
419             **                sEXTGS = 1
420             **
421             **                Used when error occurs trying to execute
422             **                ON TIMER statement: See GOTO/GOSUB
423             **
424             ** Exit:
425             **      If OFF ERROR | TIMER ---> Return through NXTSTM
426             **      IF OFFTMR and External Entry --> RTNYES
427             **      If OFF | BYE
428             **         If Program
429             **            Gosub to Deep Sleep
430             **         else
431             **            golong Power OFF
432             **
433             ** Calls:       GTMR#, GTMRA+, PUTALM, DSLEEP
434             **
435             ** Uses.......
436             **  Exclusive: A,C,DO,D1,ERRADR,TMRAD1-3,sEXTGS (S5)
437             **
438             ** Stk lvls:   OFF:    7
439             **             OFFTMR: 2
440             **
441             ** Detail:
442             **      Clear External Entry flag              (sEXTGS)
443             **      If OFF ERROR
444             **         Zero out ERRADR
445             **         Return to Run Loop through NXTSTM
446             **      If OFF TIMER
447             **         Get Timer#                          (GTMR#)
448             **         Save Timer#                         (A(S)
449             ** OFFTMR: Zero ALRM RAM for Timer             (PUTALM)
450             **         Restore Timer#
451             **         Get ON TIMER address for Timer#     (GTMRA+)
452             **         Zero appropriate ON TIMER address
453             **         If External Entry                   (sEXTGS)
454             **            RTNYES
455             **         else
```

```
                **          Return to Run Loop through NXTSTM
                **      If OFF | BYE
                **         If program
                **            Gosub to Deep Sleep/wait for Wake-up (DSLEEP)
                **            golong NXTSTM
                **         else
                **            golong Power OFF              (PWROFF)
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **   --------   ----------   ----------------------------------
                **  07/04/82   JP           Modified documentation
                **  10/04/82   JP           Add OFFTMR entry
                **
                **
                ********************************************************************
                ********************************************************************
07F86 0000 0    REL(5) =OFFDC         OFF Decompile Address
07F8B 0000 0    REL(5) =OFFP          OFF Parse Address
07F90 845  =OFF ST=0   sEXTGS         Clear External entry flag
07F93 14A       A=DAT0 B
                *
                * OFF ERROR
                *
07F96 3100      LC(2)  =tERROR        ERROR token
07F9A 966       ?A#C   B
07F9D 61        GOYES  OFF20
07F9F 1F88 6F2  D1=(5) =ERRADR        Zero ERRADR
07FA6 D2   OFF10 C=0   A
07FA8 145       DAT1=C A
07FAB 875       ?ST=1  sEXTGS         External entry ?
07FAE 00        RTNYES                Return to GOTO/GOSUB
07FB0 524       GONC   OFFEXT         Return to Run Loop
                *
                * OFF TIMER
                *   Get Timer#, Save Timer#
                *
                * OFFTMR entry
                *     A(0) = Timer#
                *     sEXTGS = 1
                *   Clear ALRM RAM for Timer#
                *     C(0) = Timer# - 1
                *     A(0-11) = Interval = 0
                *   Get ON TIMER Address
                *
07FB3 E6   OFF20 C=C+1 A              TIMER token
07FB5 966       ?A#C   B
07FB8 A2        GOYES  OFF30
07FBA 7C01      GOSUB  GTMR#          Get Timer#
07FBE D6   =OFFTMR C=A A              C(0) <-- Timer#
07FC0 814       ASRC                  Save Timer# --> A(S)
```

```
508 07FC3 CE           C=C-1  A              Alarm# = Timer# - 1
509 07FC5 AD0          A=0    M              Zero Timer interval - A(0-11)
510 07FC8 D0           A=0    A              A(S) must be preserved
511 07FCA 8F00         GOSBVL =PUTALM        Clear Alarm
          000
512 07FD1 7221         GOSUB  GTMRA+         Get Timer Address
513 07FD5 50D          GONC   OFF10          B.E.T. Go zero approp TIMER addr
514               *
515               * OFF | BYE
516               *
517 07FD8 0000         REL(5) =STOPDC        BYE Decompile
          0
518 07FDD 0000         REL(5) =RTNCC         BYE Parse
          0
519               *
520               * If not running
521               *    golong Power Off
522               * else
523               *    gosub to Deep Sleep and wait for Wake-up
524               *
525 07FE2         =BYE
526 07FE2 87D   OFF30  ?ST=1  PgmRun         Program Running ?
527 07FE5 80           GOYES  OFF40
528 07FE7 8C00         GOLONG =PWROFF        No - Goto Power Off
          00
529 07FED 8E00  OFF40  GOSUBL =DSLEEP        Go to sleep, wait for Wake-up
          00
530 07FF3 646E  OFFEXT GOTO   ONEXIT         Golong NXTSTM
```

```
                STITLE ONERR/TIMR - ON ERROR/TIMER Exec
***************************************************************
***************************************************************
**
** Name:    ONERR   -  Execute branch of ON TIMER/ERROR
** Name:(S) ONTIMR  -  Execute branch of ON TIMER/ERROR
**
** Category:   STEXEC
**
** Purpose:
**      Process ON TIMER execution
**      Process ON ERROR execution
**
**      Indicates code needed to process any statement with
**      GOTO/GOSUB that interrupts program execution and wants
**      TRACE.  This code must be duplicated
**
**      The main difference is sXWORD should be set before the
**      call to GOTO+.  This guarantees that all line# references
**      will be searched for, incase the reference was never clea
**      due to the LEX file being missing when clearing reference
**
**      Example statement: ON INTR GOTO|GOSUB <stmt id>
**
** Entry:
**      C  @ GOTO | GOSUB of statement
**      For ONTIMR: RSTK = Next Stmt Address
**                  sONTMR = 1  (S6)
**                  sONERR = 0  (S4)
**                  A(S) = Timer #
**      Duplicate this code for
**      External Statement w/GOTO or GOSUB with interupt
**      Make sure sXWORD is set before jumping to GOTO+
**      This code will TRACE properly
**
** Exit:
**      Through GOTO+ to execute GOTO | GOSUB
**      RSTK = Next Stmt address
**      sEXTGS = 1
**      If ONTIMER: sONTMR = 1
**                  R3(S)  = Timer#
**      If ONERROR: sONERR = 1
**
** Calls:      TRFCK-,TRFROM,UPDPC,TRTO*,RACTMI,LNSKP-
**
** Uses.......
**  Exclusive: sGOSUB(S3),sEXTGS(S5),sONERR(S4),S6,S9,R1,R2,R3
**
**  sGOSUB    S3 = GOSUB flag
**  sONTMR    S6   = ON TIMER statement
**  sEXTGS    S5   = External Entry flag for GOTO+
**  sONERR    S4   = ON ERROR statement
**  sXWORD    S9   = XWORD flag for searching for GOTOs
**            RSTK = Return Address if GOSUB & ON TIMER
**            R2   = Position @ <lineno> | <label> in stmt
```

```
586            **                    Saved DO
587            **              R3(S)= Timer#  (if ON TIMER)
588            **              A(S) = Timer#  (if ON TIMER)
589            **
590            **  RACTMI     uses R0,R1,R3,
591            **
592            ** Stk lvls:    <= 7  (statement execute)
593            **
594            ** Detail:
595            **
596            ** ONERR:    Clear status
597            **           Set ON ERROR flag
598            **           Compute next statement return addr(LNSKP-)
599            **           Save on stack
600            ** ONTIMR:   Set External Entry flag          (sEXTGS)
601            **           Set D0 = C  (position within ON statement)
602            **           Read and skip over token
603            **           Save D0 in R2
604            **           Save Timer# in R3(S)
605            **           Set GOSUB flag
606            **           If GOTO
607            **              Clear GOSUB flag
608            **              If ON TIMER
609            **                 Reactivate timer          (RACTMI)
610            **                 Resave timer#             (R3(S))
611            **           If trace needed                 (TRFCK-)
612            **              Trace FROM line#             (TRFROM)
613            **              Restore D0
614            **           Update PC address to point to ON stmt
615            **              If trace needed              (TRFCK-)
616            **                 Trace TO line#            (TRTO*)
617            **           Restore D0
618            ** ONGTGB:   Clear XWORD flag                (sXWORD)
619            **           go execute GOTO | GOSUB of statement
620            **
621            ** History:
622            **
623            **    Date     Programmer          Modification
624            **  --------   ----------   ---------------------------------
625            **  07/04/82   JP           Modified documentation
626            **  09/28/82   JP           Changed ON TIMER implementation
627            **  11/28/82   JP           Changed interface to GOTO/GOSUB
628            **  12/08/82   JP           Fixed Timer# destroy by TRACE
629            **  02/11/83   JP           Clear sXWORD before GOTO+ jump
630            **  03/08/83   JP           Removed sEXTGS set, clear @ ONERR
631            **  03/31/83   JP           Compute Rtnadr for ON ERROR
632            **  03/31/83   JP           Always update PCADRR @ ON stmt
633            **
634            ****************************************************************
635            ****************************************************************
636 07FF7 08   =ONERR  CLRST
637 07FF9 854          ST=1   sONERR         ON ERROR statement flag.
638            *
639            * Return address must be computed and pushed on stack
640            *   If TRACING
```

```
                *      Trace will change the PCADDR to point to the ON ERROR
                *      Therefore, the LNSKP computation in GOTO/GOSUB is too late
                * Preserve C on entry: points @ GOTO/GOSUB of ON ERROR statement
                *
07FFC 8E00 00           GOSUBL =LNSKP-          Compute Return address
08002 DE                ACEX   A                C<-- Return addr; A<-- C on entry
08004 06                RSTK=C                  Push return address on stack
08006 D6                C=A    A                C <-- C on entry (@GOTO/GOSUB)
                *
                * ON TIMER execute entry
                *    C= Address into ON statement @ GOTO | GOSUB
                *    RSTK = Next statement address
                *    A(S) = Timer #
                * Clear RESTORE statement flag
                * Set External Entry flag for GOTO/GOSUB
                *    Indicates that Return address is on STACK
                *    Avoids Return address calculation @ PCADDR
                * Set D0= Position in ON statement past GOTO | GOSUB
                * Save D0   (must use R2, since RACTMI uses R0,R1 & R3
                * Save Timer# in R3   (TRACE destroys A(S))
                * Set GOSUB flag
                *
08008 84A  =ONTIMR ST=0   sRESTR          Clear RESTORE flag
0800B 855          ST=1   sEXTGS          Set External Entry flag
0800E 134          D0=C
08011 14A          A=DAT0 B
08014 161          D0=D0+ 2               Move past token
08017 136          CD0EX                  Save D0 in R2
0801A 10A          R2=C
0801D 103          R3=A                   Save Timer#  (A(S))
08020 853          ST=1   sGOSUB          Set GOSUB flag
08023 20           P=     0               Reset P
08025 300          LC(1)  =tGOTO          GOTO token low nibble
08028 906          ?A#C   P               GOSUB ?
0802B 71           GOYES  ONTM20
                *
                * Clear GOSUB flag
                * If ON TIMER GOTO
                *    Reactivate Timer Interval for Corresponding timer #
                *    Restore Timer#-1 to A(S)
                *      On return from RACTMI, R3(S) = Timer# - 1
                *    Resave Timer# into R3
                *
0802D 843          ST=0   sGOSUB          Clear GOSUB flag
08030 866          ?ST=0  sONTMR          not ON TIMER ?
08033 F0           GOYES  ONTM20
08035 7A30         GOSUB  RACTMI          Reactivate timer interval
08039 113          A=R3                   Timer# - 1
0803C B44          A=A+1  S               Timer#
0803F 103          R3=A                   Resave Timer# cus of TRACE
                *
                * Save D0 in R2
                * If trace needed
                *    Trace FROM line#
```

```
695                *
696 08042 7000   ONTM20 GOSUB  =TrfCk-        Check if need to trace
697 08046 480           GOC    ONTM25
698 08049 8E00          GOSUBL =TRFROM        Trace the FROM line #
          00
699                *
700                * Update PC Address to ON statment
701                *   TRACE needs PC @ ON statement for Line#
702                *   GOTO/GOSUB needs PC @ ON statement incase of Error
703                *
704 0804F 11A    ONTM25 C=R2                  Restore D0
705 08052 134           D0=C
706 08055 7000          GOSUB  =UPDPC         Update PC to ON statement
707                *
708                * If trace needed
709                *    Trace TO line#
710                *
711 08059 7000          GOSUB  =TrfCk-        Check if need to trace
712 0805D 480           GOC    ONTM30
713 08060 8E00          GOSUBL =TRTO*         Trace the TO line #
          00
714                *
715                * Restore D0 past GOTO | GOSUB of statement
716                * Clear sXWORD flag
717                *   At this point:
718                *      sEXTGS set if ON TIMER or External Entry
719                *      All others have cleared: ONERR,ON
720                * go execute GOTO | GOSUB of statement
721                *
722 08066 11A    ONTM30 C=R2                  Restore D0
723 08069 134           D0=C
724 0806C 849    ONGTGB ST=0   sXWORD         Clear XWORD flag
725 0806F 6000          GOTO   =GOTO+
```

```
726                 $TITLE RACTMI - Reactivate Timer Interval
727             ****************************************************************
728             ****************************************************************
729             **
730             ** Name:    RACTMI  -  Reactivate ON TIMER Interval
731             **
732             ** Category:   TIME
733             **
734             ** Purpose:
735             **      Reactivate timer interval for a specific timer# 1-3
736             **
737             ** Entry:
738             **      P      =  0
739             **      A(S)   =  Timer#
740             **
741             ** Exit:
742             **     Carry Clear
743             **     R3(S) = Timer# - 1
744             **     P = 0
745             **
746             ** Calls:      RCTTM1,TMIADR
747             **
748             ** Uses.......
749             **  Exclusive: A,C,P,D1
750             **  Inclusive: A-D,P,D0,D1,R0,R1,R3,S0-S7
751             **
752             ** Stk lvls:   4 ( to preserve Status)
753             **
754             ** Detail:
755             **      Timer Intervals:
756             **        TMRIN1  8 nibs
757             **        TMRIN2  8 nibs
758             **        TMRIN3  8 nibs
759             **
760             **      Entry Conditions to RCTTM1
761             **        C(S)   = Timer#
762             **        A(0-11)= Timer interval in 1/512 accuracy
763             **
764             ** Algorithm:
765             **      Preserve Status                        (RSTK)
766             **      Save Timer#                            (C(S))
767             **      Multiply Timer# by 8 for offset into Interval area
768             **      Add offset to Interval start - 8
769             **      Read Timer interval
770             **      Shift for 1/512 accuracy
771             **      Reactivate Timer interval              (RCTTM1)
772             **      Restore Status                         (RSTK)
773             **
774             ** History:
775             **
776             **    Date     Programmer             Modification
777             **  --------   ----------   ---------------------------------
778             **  09/28/82   JP           Added code
779             **
780             ****************************************************************
```

```
781             ****************************************************************
782 08073 0B    =RACTMI CSTEX
783 08075 06            RSTK=C                  Save Status
784 08077 AC6           C=A     S               Save Timer#
785 0807A D0            A=0     A
786 0807C 810           ASLC                    Move Timer # to A(0)
787 0807F C4            A=A+A   A               Timer# * 2
788 08081 C4            A=A+A   A               Timer# * 4
789 08083 D6            C=A     A
790 08085 7060          GOSUB   TMIADR
791 08089 AF0           A=0     W               Clear top part of A
792 0808C 15B7          A=DAT1  8               Read Timer interval
793 08090 BF0           ASL     W               Shift for 1/512 accuracy
794 08093 8F00          GOSBVL  =RCTTM1         Reactivate timer
              000
795 0809A 07            C=RSTK                  Restore status
796 0809C 0B            CSTEX
797 0809E 03            RTNCC
```

```
                        STITLE RNDEXP - Eval expr, rnd, to HEX
                ********************************************************************
                ********************************************************************
                **
                ** Name:    RNDEXP  -  Evaluate, Round, Convert Expression to Hex
                **
                ** Category:   EXCUTL
                **
                ** Purpose:
                **      Evaluate expression, round and convert to Hex
                **      Only a real, non-zero, positive, finite argument accepted
                **      Error exits: Array, complex, Nan, infinity, Zero
                **
                ** Entry:
                **      DO @ expression to evaluate
                **
                ** Exit:
                **      Carry Set
                **        A(A) = Rounded, hexadecimal value
                **        DO = past expression
                **
                **      Error Exits
                **        eIVARG  - Expression out of range, neg number, zero
                **        eVARTY  - Array or complex  (from ARGSTA)
                **        eIVARG  - NaN
                **        ePROJ   - Proj Inf
                **
                ** Calls:      EXPEXC, FLTDH, ARGSTA
                **
                ** Uses:
                **  Exclusive: A,B,DO
                **  Inclusive: A, Expression Execute usage:
                **                Everything but Stmt scratch
                **
                ** Stk lvls:   5
                **
                ** Detail:
                **      Evaluate expression                 (EXPEXC)
                **      Standardize argument                (ARGSTA)
                **        Error exits - Array,complex
                **      Restore to 12 digit form
                **      Round and convert to hex            (FLTDH)
                **      If carry clear or argument=0
                **         Error Exit ---> "Invalid Arg"
                **      else
                **         RTN
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **   --------   ----------   ----------------------------------
                **  07/04/82   JP           Modified documentation
                **  09/03/82   JP           Interfaced to ARGSTD
                **  10/14/82   JP           Interfaced to ARGSTA
                **  02/11/83   JP           Removed EXPER1, ExpExc
```

```
853                ** 05/11/83   JP          Check and errror is argument=0
854                **
855                ************************************************************************
856                ************************************************************************
857 080A0 8E00 =RNDEXP GOSUBL =EXPEXC        Evaluate expression
          00
858 080A6 8E00         GOSUBL =ARGSTA        Standardize argument
          00
859 080AC 8E00         GOSUBL =fltdh         Round, convert to hex
          00
860 080B2 570          GONC   EXPER2         Invalid arg:neg or out of range
861 080B5 8AC          ?A#0   A              Non-zero argument ?
862 080B8 00           RTNYES
863                *
864                * Error Exits
865                *
866 080BA 8C00 =EXPER2 GOLONG =ARGERR        "Invalid Arg"
          00
867                *
868 080C0 3100  XWDERR LC(2)  =eXWORD        XWORD Missing
869 080C4 8C00  Mferr  GOLONG =MFERR
          00
```

```
870                 STITLE Get Timer#
871             ****************************************************************
872             ****************************************************************
873             **
874             ** Name:    GTMR#   -  Get Timer#
875             **
876             ** Category:   TIME
877             **
878             ** Purpose:
879             **      Evaluate <timerno> expression, validate it
880             **      Poll if Timer# > 3 to allow expansion in LEX File
881             **
882             ** Entry:
883             **      DO @ Timer# expression
884             **
885             ** Exit:
886             **      If returns:
887             **         A(A) = Timer# between 1 and 3
888             **
889             **      Error exits:
890             **         eIVARG  Invalid Arg
891             **                 Timer# <= 0 or Timer# > 3
892             **         eDATTY  Illegal Data Type
893             **                 Timer# = Complex
894             **         eMEM    Insufficient Memory to Poll w/ Timer# > 3
895             **
896             **      External exit:
897             **         Response to pTIMR# to take over ON/OFF TIMER stmt
898             **
899             ** Calls:      RNDEXP
900             **
901             ** Uses.......
902             **  Exclusive: A(A),C(A),DO
903             **  Inclusive: EXPEXC usage: everything but Stmt Scratch
904             **
905             ** Stk lvls:   6
906             **
907             ** Algorithm:
908             **      Skip over TIMER token
909             **      Evaluate Timer expression
910             **      Error exit if Complex or Negative
911             **      Round and convert to HEX
912             **      If Timer# > 3
913             **         Poll for Timer#
914             **         If carry set
915             **            "eMEM" Error from Poll
916             **         else
917             **            "eIVARG" Error
918             **
919             ** History:
920             **
921             **      Date     Programmer          Modification
922             **    --------   ----------   ---------------------------------
923             **    07/04/82   JP           Modified documentation
924             **    01/19/83   JP           Added pTIMR# Poll
```

```
925           **   05/11/83   JP          RNDEXP errors if arg=0
926           **
927           ********************************************************************
928           ********************************************************************
929           ********************************************************************
930           ********************************************************************
931           **
932           ** Name:(S) pTIMR#  -  Poll Timer# > 3 for ON/OFF TIMER
933           **
934           ** Category:   POLL
935           **
936           ** Type:       POLL
937           **
938           ** Purpose:
939           **      Poll on Timer# > 3 for ON TIMER and OFF TIMER statements
940           **      Allows Lex File to extend these statements to more than
941           **        3 timers
942           **
943           ** Should poll be "Handled" (return with XM=0)?:
944           **      No - If this poll is handled
945           **             Return is through NXTSTM to continue
946           **             statement/program execution.
947           **
948           ** Meaning of "Handling" Poll (what does code do if handled?):
949           **      For ON TIMER:  Set up the bookkeeping
950           **                     Activate the appropriate timer
951           **      For OFF TIMER: Deactivate the appropriate timer
952           **
953           ** Entry conditions for handler
954           **      B[A] = Poll number (pTIMR#)
955           **      HEX mode.
956           **      P=0.
957           **
958           **      A(A)   = Timer# > 3 in HEX
959           **      DO     @ past Timer# expression
960           **               If ON TIMER:  DO @ tCOMMA
961           **                             Comma before timer interval
962           **               If OFF TIMER: DO @ Remark or tEOL or t@
963           **
964           **      PCADDR @ Statement length byte for statement
965           **               (PCADDR) + 2 @ tON or tOFF
966           **
967           ** Normal exit conditions from handler if handled (ST, RAM,
968           ** registers, etc.):
969           **      Return through NXTSTM to continue statement execution
970           **      HEX mode.
971           **      NOTE:
972           **        If binary code invokes BASIC through CALL:
973           **           PCADDR must be saved on the GOSUB stack before CALL
974           **             Call PSHUPD
975           **           and restored before NXTSTM is jumped to
976           **             Call POPUPD
977           **
978           ** Normal exit conditions from handler if not handled:
979           **
```

```
980             **      Carry clear
981             **      HEX mode.
982             **      XM=1.
983             **
984             ** Error exit conditions from handler:
985             **      There is no error return from this poll
986             **
987             ** Available subroutine levels:
988             **      --POLL handler is one level shallower than caller--
989             **      6 levels available
990             **
991             ** What registers/RAM may be used if handled?:
992             **      --A-D, D0, D1, P always available--
993             **      This is a Statement Execute
994             **      All RAM and registers allowed during Statement Execute
995             **
996             ** What registers/RAM may be used if not handled?:
997             **      --A-D, D0, D1, P always available
998             **
999             ** What registers/RAM may be used if error exit:
1000            **      No error return allowed
1001            **
1002            ** Special considerations :
1003            **
1004            **     Tokenized form of statements:
1005            **     ON TIMER:
1006            **       Stmt Length, tON, Timer expression, tCOMMA,
1007            **       Interval expression, tGOTO or tGOSUB, statement ident.
1008            **
1009            **     OFF TIMER:
1010            **       Stmt Length, tOFF, Timer expression
1011            **
1012            **     To service a Timer when it goes off:
1013            **        Respond to pSREQ poll to set sExcept
1014            **            to indicate an Exception has occured
1015            **        Respond to pExcept to actually service the timer
1016            **
1017            **     To execute Timer branch:
1018            **        Use GOTO+ entry point after:
1019            **          Setting sGOSUB (S3) if GOSUB
1020            **          Reactivating Timer if GOTO
1021            **          Setting sEXTGS (S5) to indicate external entry
1022            **          Setting sXWORD (S9) for line# searching
1023            **          Pushing Return Address (from Timer interrupt)
1024            **            on stack
1025            **          Tracing FROM line
1026            **          (see ONTIMR for parallel code)
1027            **
1028            **
1029            ** Envisioned application(s):
1030            **      Extending Timers to an infinite number with a Lex file
1031            **      that allocates an I/O buffer to keep track of pending
1032            **      timers.
1033            **
1034            ** History:
```

```
1035            **
1036            **    Date      Programmer              Modification
1037            **  --------    ----------   ---------------------------------
1038            **  01/19/83    JP           Added Poll
1039            **  04/19/83    JP           Revised Poll documentation
1040            **
1041            ********************************************************************
1042            ********************************************************************
1043 080CA 161  =GTMR#  DO=DO+ 2              Skip over TIMER token
1044 080CD 7FCF         GOSUB  RNDEXP         Evaluate & round expression
1045 080D1 D2           C=0    A
1046 080D3 303          LCHEX  3
1047 080D6 8B6          ?A>C   A              Timer# > 3 ?
1048 080D9 40           GOYES  GTMR10
1049 080DB 03           RTNCC
1050 080DD 7073  GTMR10 GOSUB  Poll
1051 080E1 B3           CON(2) =pTIMR#        Poll for Timer# > 3
1052 080E3 40E          GOC    Mferr          Memory Error in Poll
1053 080E6 53D          GONC   EXPER2         Invalid Arg
```

```
1054                    STITLE Get ON TIMER address Pointer
1055                ********************************************************************
1056                ********************************************************************
1057                **
1058                ** Name:     GTMRAD  -  Get ON TIMER Address pointer
1059                ** Name:     GTMRA+  -  Get ON TIMER Address pointer
1060                ** Name:     TMIADR  -  Get TIMER Interval Address pointer
1061                **
1062                ** Category:   TIME
1063                **
1064                ** Purpose:
1065                **       Compute ON TIMER address pointer based on Timer#
1066                **       Compute ON TIMER Interval address pointer
1067                **
1068                ** Entry:
1069                **       GTMRAD:  C(A) = Timer#  (1-3)
1070                **       GTMRA+:  A(S) = Timer#  (1-3)
1071                **       TMIADR:  C(A) = Timer# * 4
1072                **
1073                ** Exit:
1074                **       GTMRAD/GTMRA+:
1075                **         D1 @ ON TIMER address for Timer#
1076                **         C(A) = Timer# * 4
1077                **
1078                **         If GTMRA+ entry:
1079                **            A(S) = Timer#
1080                **
1081                **       TMIADR:
1082                **         D1 @ ON TIMER Interval address for Timer#
1083                **
1084                ** Calls:       None
1085                **
1086                ** Uses.....
1087                **  Exclusive: A(A),A(S),C(A),C(S),D1
1088                **  Inclusive: A(A),A(S),C(A),C(S),D1
1089                **
1090                ** Stk lvls:   0
1091                **
1092                ********************************************************************
1093                ********************************************************************
1094 080E9 1FE9 =TMIADR D1=(5) (=TMRIN1)-8    Timer Interval start - 8
          6F2
1095 080F0 133          AD1EX
1096 080F3 6910         GOTO   GTMRA1
1097 080F7 AC6  =GTMRA+ C=A    S              Timer#
1098 080FA D2           C=0    A
1099 080FC 812          CSLC                  Move Timer# to C(A)
1100 080FF 1F29 =GTMRAD D1=(5) (=TMRAD1)-5    Start of Timer addresses -5
          6F2
1101 08106 133          AD1EX
1102 08109 CA           A=A+C  A              Start address + Timer#
1103 0810B C6           C=C+C  A              Timer# * 2
1104 0810D C6    GTMRA1 C=C+C  A              Timer# * 4
1105 0810F CA           A=A+C  A              Positioned  @ Timer address
1106 08111 131          D1=A                  ON TIMER address pointer
```

```
1107 08114 03          RTNCC
```

```
1108                    STITLE XWORD - Calc, Jump to XWORD exec
1109                ********************************************************************
1110                ********************************************************************
1111                **
1112                ** Name:    XWORD   -  Calculate Execution Addr & Jump
1113                **
1114                ** Category:   STEXEC
1115                **
1116                ** Purpose:
1117                **      Calculate Execution Address of a particular XWORD
1118                **      Jump to Execution routine
1119                **
1120                ** Entry:
1121                **      D0 @ LEX ID# for XWORD from Run Loop
1122                **
1123                ** Exit:
1124                **      Through appropriate execution routine
1125                **
1126                **      Error Exit - If Execution Addr not found
1127                **                   "XWORD Missing"
1128                **
1129                ** Calls:      XMTADR, EXCAD+
1130                **
1131                ** Uses.......
1132                **  Exclusive: A(A),C(A),D0,+1 lvl subroutine stack
1133                **  Inclusive: A(A),B(A),C(A),D0,R1,+1 lvl subroutine stack
1134                **
1135                ** Stk lvls:   2
1136                **
1137                ** Algorithm:
1138                **
1139                **      Read LEX ID, Entry#     (D0 --> A)
1140                **      Find MAINT address      (XMTADR)
1141                **      If address not found
1142                **         golong MFERR   ("XWORD Missing")
1143                **      Increment postion within MAINT to Execution Addr entry
1144                **      Skip LEX ID, Entry#
1145                **      Move entry# to A(A)
1146                **      Calculate Execution Address  (EXCAD+)
1147                **      Place address on stack
1148                **      Jump to Execution routine
1149                **
1150                ** History:
1151                **
1152                **    Date      Programmer              Modification
1153                **  --------   ----------   ----------------------------------
1154                **  07/04/82   JP           Modified documentation
1155                **
1156                **
1157                ********************************************************************
1158                ********************************************************************
1159 08116 142  =XWORD  A=DAT0 A              Read ROM ID, Entry#
1160 08119 7610         GOSUB  =XMTADR        Find MAINT address
1161 0811D 42A          GOC    XWDERR         Address not found
1162 08120 E6           C=C+1  A              Add offset to Execution addr
```

```
1163 08122 E6          C=C+1  A              within table
1164 08124 E6          C=C+1  A
1165 08126 163         DO=DO+ 4              Skip ROM ID, Entry#
1166 08129 D4          A=B    A              Move Entry # to A
1167 0812B 7205        GOSUB  EXCAD+         Calculate Execution Address
1168 0812F 06          RSTK=C
1169 08131 01          RTN                   Jump to Execution Routine
```

```
1170                    STITLE Find XWORD MAINT Addr
1171            ********************************************************************
1172            ********************************************************************
1173            **
1174            ** Name:(S) XMTADR  -  Get XWORD Main Table Address
1175            **
1176            ** Category:   ADDCAL
1177            **
1178            ** Purpose:
1179            **      Find & Read XWORD MAINT Address
1180            **
1181            ** Entry:
1182            **      A(B)   = LEX ID
1183            **      A(2,3) = Entry #
1184            **
1185            ** Exit:
1186            **      Carry clear
1187            **         C   = MAINT address for XWORD
1188            **         B(A)= Relative Entry # for LEX ID with B(2-4) = 0
1189            **         A(B)= Actual Entry #
1190            **
1191            **      Carry set
1192            **        LEX ID not found
1193            **
1194            **      D1 preserved
1195            **
1196            ** Calls:      LXFND, RANGE
1197            **
1198            ** Uses.....
1199            **  Exclusive: A(A),B(A),C(A),R1
1200            **             R1 = Preserved D1, RSTK holds LEX ID, Entry#
1201            **
1202            **  Inclusive: A(A),B(A),C(A),R1
1203            **
1204            ** Stk lvls:   1
1205            **
1206            ** Algorithm:
1207            **
1208            **      Find Main Table Address for ROM ID
1209            **        Save LEX ID, Entry#  (B)
1210            **        Save D1              (R1)
1211            **        Find LEX Table Buffer    (LXFND)
1212            **        If Buffer not found --> goto 1  (return, carry set)
1213            **        Save LEX ID, Entry#  (RSTK)
1214            **        Repeat until (LEX ID = 0)
1215            **           Read LEX ID in table
1216            **           If IDs match
1217            **              Pop Lex ID, Entry # off stack
1218            **              If Entry# within Range for LEX ID     (RANGE)
1219            **                 Shift Entry# to B(B), Zero B(XS) field
1220            **                 Compute Relative entry #
1221            **                 Read Main Table address --> C
1222            **                 Restore D1
1223            **                 RTNCC
1224            **              Restore LEX ID,Entry# to B(A)
```

```
1225                **          Skip to next entry
1226                **    0:  Pop LEX ID, Entry # off stack
1227                **    1:  Restore D1
1228                **        RTNSC (not found)
1229                **
1230                ** History:
1231                **
1232                **    Date      Programmer            Modification
1233                **  --------   ----------   ----------------------------------
1234                **  07/04/82   JP           Modified documentation
1235                **  11/01/82   JP           Interfaced to New Lex File format
1236                **  03/28/83   JP           Save LexID, Entry # on Stack
1237                **  04/28/83   JP           Restore Entry# to A(B)
1238                **
1239                ********************************************************************
1240                ********************************************************************
1241 08133 D8     =XMTADR B=A    A              Save ROM ID, Entry#
1242 08135 133          AD1EX
1243 08138 101          R1=A                  Save D1
1244                *
1245                * Find Start of LEX Buffer
1246                *
1247 0813B 8E00         GOSUBL =LXFND         Find Start of Lex Table Buffer
           00
1248 08141 584          GONC   XMTA30         Lex Buffer not found,Set carry, Rtn
1249                *
1250                * Save Lex ID and Entry # on stack
1251                *
1252 08144 D9           C=B    A              Save Lex ID and Entry#
1253 08146 06           RSTK=C
1254                *
1255                * Read Lex IDs until ID=0 or Match
1256                *
1257 08148 15F5  XMTA10 C=DAT1 (1LXID)+(1LXTKR)  Read Lex Id, Token Range
1258 0814C 96A          ?C=0   B              Lex ID# = 0 ?
1259 0814F 93           GOYES  XMTA25
1260 08151 965          ?B#C   B              IDs Don't Match ?
1261 08154 E2           GOYES  XMTA20
1262                *
1263                * Shift off Lex Id to get Token range in C(0-3)
1264                * Shift Entry # into B(B) and restore into A(B)
1265                * Compute Relative Entry#
1266                * Check if desired Entry# within range of Lex File
1267                * Pop Lex ID, Entry# off stack
1268                *
1269 08156 BF6            CSR    W
1270 08159 F6           CSR    A              Shift off Lex ID
1271 0815B F5           BSR    A              Shift entry# to B(B)
1272 0815D F5           BSR    A
1273 0815F AA1          B=0    XS             Zero high nibs
1274 08162 D4           A=B    A              Move Actual entry# to A(B)
1275 08164 B61          B=B-C  B              Relative Entry#
1276 08167 8E00         GOSUBL =Range         Check if entry# within range of Lex
           00
1277 0816D 07           C=RSTK                Pop Lex ID, Entry#
```

```
1278 0816F 4E0          GOC    XMTA15          Not within range, restore B(A),cont
1279              *
1280              * Lex ID and Entry# within range found
1281              *   Read Main table address
1282              *   Restore D1 and RTNCC
1283              *
1284 08172 175          D1=D1+ (1LXID)+(1LXTKR)  Skip LEX ID, Token Range
1285 08175 147          C=DAT1 A                Read Main table address
1286 08178 7E00         GOSUB  XMTA30           Restore D1
1287 0817C 03           RTNCC
1288              *
1289              * Restore Lex ID, Entry# to B(A)
1290              * Skip to next table entry
1291              * Check if end of table
1292              *
1293 0817E D5    XMTA15 B=C    A                Restore Lex ID, Entry#
1294 08180 06           RSTK=C                  Resave Lex ID, Entry#
1295 08182 17A   XMTA20 D1=D1+ 1LXENT           Skip to next table entry
1296 08185 52C          GONC   XMTA10           B.E.T.
1297              *
1298              * Pop Lex ID, Entry # off stack
1299              * Lex ID = 00 - ROM ID not found
1300              *
1301              * If Address found (GOSUB XMTA30 above)
1302              *    Need to preserve A(B) will restoring D1
1303              *
1304 08188 07    XMTA25 C=RSTK                  Pop Lex ID, Entry#
1305 0818A 121   XMTA30 AR1EX                  A <-- old D1; R1 <-- A
1306 0818D 131          D1=A                    D1 <-- old D1
1307 08190 111          A=R1                    A <-- old A
1308 08193 02           RTNSC                   Set carry on exit
```

```
1309                    STITLE Main Table Entry Addr Calc
1310                ********************************************************************
1311                ********************************************************************
1312                **
1313                ** Name:(S) MTADDR  -  Calc Main Table Address for Token
1314                ** Name:(S) MTADR+  -  Calc Main Table Address for Token
1315                **
1316                ** Category:   ADDCAL
1317                **
1318                ** Purpose:
1319                **      Calculates address of Main Table entry for token
1320                **
1321                ** Entry:
1322                **      MTADDR:  A(B) = Token to be looked up
1323                **               Loads C with Mainframe MAINT
1324                **
1325                **      MTADR+:  B(A) = Token to be looked up
1326                **               C(A) = Main table address
1327                **
1328                ** Exit:
1329                **     D1 contains main table entry address for token
1330                **     C(A) contains value of D1 at time of call
1331                **
1332                ** Calls:      None
1333                **
1334                ** Uses.......
1335                **  Exclusive: B(B),C(A),A(A),D1
1336                **  Inclusive: B(B),C(A),A(A),D1
1337                **
1338                ** Stk lvls:   0
1339                **
1340                ** Detail:
1341                **      Multiplies token number by length of Main Table entry
1342                **
1343                ** History:
1344                **
1345                **    Date     Programmer              Modification
1346                **  --------   ----------   ----------------------------------
1347                **  07/04/82   JP           Modified documentation
1348                **
1349                **
1350                ********************************************************************
1351                ********************************************************************
1352 08195 3400 =MTADDR LC(5)  =MAINT         Main table offset
          000
1353 0819C D1          B=0    A
1354 0819E AE8         B=A    B              Copy token to B(A)
1355 081A1 C9   =MTADR+ C=C+B  A
1356 081A3 C5          B=B+B  A
1357 081A5 C5          B=B+B  A
1358 081A7 C5          B=B+B  A              Multiply index by 9
1359 081A9 C9          C=C+B  A              Add offset
1360 081AB 137         CD1EX                 Swap address into D1
1361 081AE 01          RTN
```

```
1362                    STITLE COPY Execute
1363            ********************************************************************
1364            ********************************************************************
1365            **
1366            ** Name:    COPY    -  Statement Execute
1367            ** Name:    RENAME  -  Statement Execute - Part 1
1368            **
1369            ** Category:   STEXEC
1370            **
1371            ** Purpose:
1372            **      Execute COPY Statement
1373            **      Execute start of RENAME statement
1374            **
1375            ** Entry:
1376            **      D0 past COPY token
1377            **
1378            ** Exit:
1379            **      Through NXTSTM if no Errors
1380            **      Through MFERR if Error Return from COPYu
1381            **
1382            **      Error Returns:
1383            **
1384            **      eMEM     - Saving file information
1385            **                 Creating destination file
1386            **      eFSPEC   - Rtn from File Spec Exec
1387            **                 No response to COPY Poll
1388            **                 Non Mainframe for CARD
1389            **      eFnFND   - Source file
1390            **      eFPROT   - Private source file
1391            **      eFEXST   - Destination file
1392            **      eFTYPE   - Non KEYS file for KEYS copy
1393            **
1394            ** Calls:      ALINFO,DEFFIL,tKYSck,FSPECx,SVINFO,COPYu
1395            **
1396            ** Uses.......
1397            **  Exclusive: A-D,D0,D1,SAVSTK (50 nibs),R0,R1,R2,
1398            **             S0-S7,S8,S9,S12,S-R0-0 to S-R1-3
1399            **             4 levels of RSTKBF  (see COPYu usage)
1400            **
1401            **  Inclusive:
1402            **
1403            **      sRENAM = RENAME execute flag       (S6)
1404            **      sDEST  = Destination Execute flag  (S3)
1405            **      sREADI = Read file information     (S4)
1406            **      sKEYS  = COPY to KEYS              (S5)
1407            **      sPCRD  = Private CARD              (S8)
1408            **          D  = Device information
1409            **               D(0) = Device Type
1410            **                      0 = No device
1411            **               dMAIN  1 = :MAIN
1412            **               dPORT  2 = :PORT
1413            **                          D(1,2) = Extender#, Port#
1414            **                                 = FF if all ports
1415            **               dCARD  3 = CARD
1416            **               dPCRD  4 = :PCRD
```

```
1417          **                    >= 8 = PIL / Non-mainframe Device
1418          **      sCARDC = CARD Copy                  (S8)
1419          **
1420          **         R1  = Destination file start     (CREATF)
1421          **         R3  = Start of source file
1422          **
1423          **      SAVSTK- 5= Source Device Information
1424          **      SAVSTK-25= Source Filename
1425          **      SAVSTK-30= Destination Device Information
1426          **      SAVSTK-50= Destination Filename
1427          **
1428          **      POLL   uses B,C,AVMEME,XM
1429          **      FSPECx uses A-D,D0,D1,R0,R1,S6,S8,S-R0-0 to S-R1-3
1430          **      RDHDR1 uses A,S9,D1
1431          **      FINDF  use  A-D,D0,D1,S6,S8,S9,R2,R3
1432          **      MOVEU0 uses A,C,D0,D1,P
1433          **      RDINF0
1434          **      SVINF0 uses A,C,D,S3,S4
1435          **      GETPR1 uses A,C,D1
1436          **      CREATF uses A-D,D0,D1,R0,R1
1437          **      tKYSck uses A,C,D(S)
1438          **      LEXBF+ uses A-D,R1
1439          **
1440          ** Stk lvls:   7
1441          **
1442          ** Detail:
1443          **      COPY [ <file spec> | CARD |  KEYS ]
1444          **           [ TO [ <file spec> | CARD | KEYS ] ]
1445          **
1446          ** Algorithm:
1447          **
1448          ** RENAME:
1449          **      Set Rename flag
1450          ** COPY:
1451          **      Set No Leeway check for Save Area allocation (P=1)
1452          **      Allocate Save Area for File informaiton   (SALLOC)
1453          **      If not enough memory for file information
1454          **         Error Exit                             (eMEM)
1455          **  1:  Clear Device information                  (D(W))
1456          **      Blank fill Chars 9-10 of Filename in      (R0)
1457          **      Read next token
1458          **      If TO token
1459          **            Skip token
1460          **            If Destination
1461          **               File spec or Keyword must follow (goto 3)
1462          **  2:        Get current filename and device     (DEFFIL)
1463          **            Shift Device to D(A)                (goto 5)
1464          **      If EOL
1465          **         If Source
1466          **            Get current filename and device     (goto 2)
1467          **         else
1468          **            Filename <-- 0 ; Device = 0         (goto 4)
1469          **  3:  If CARD
1470          **         Device <-- CARD
1471          **         Skip token
```

```
1472                ** 4:     Filename <-- 0
1473                **      If KEYS                                  (tKYSck)
1474                **         Filename <-- "keys"
1475                **      else
1476                **         Execute File Specification           (FSPECx)
1477                **         If error  (Carry set)
1478                ** CPYERX:     golong BSERR
1479                **  5:     Shift Device #                        (D(0))
1480                **      Save File Information                    (SVINFO)
1481                **      If Source                                (sDEST=0)
1482                **         Set Destination flag
1483                **         go decifer Destination information    (goto 1)
1484                **      If RENAME
1485                **         go Execute Rename                     (RNM010)
1486                **      else
1487                **         go Copy                               (COPYu)
1488                **      If error return                          (carry set)
1489                **         goto CPYERX                           (BSERR)
1490                **      else
1491                **         golong NXTSTM
1492                **
1493                ** Note:
1494                **    The save area allocation for COPY will NEVER use Leeway
1495                **    Checking.  This allows COPY TO External Device to work
1496                **    when there is no memory left in the machine.
1497                **
1498                ** History:
1499                **
1500                **    Date     Programmer          Modification
1501                **   --------  ----------  ----------------------------------
1502                **   07/04/82  JP          Modified documentation
1503                **   10/18/82  JP          Remove PCRD as acceptable FSpec
1504                **   01/11/83  JP          Rewrote for corrected Parse routine
1505                **   01/23/83  JP          Blank fill RO, No device for KEYS
1506                **   02/03/83  JP          Removed NO device for KEYS
1507                **
1508                ************************************************************************
1509                ************************************************************************
1510                *
1511 081B0 0000         REL(5) =RENMDC
          0
1512 081B5 0000         REL(5) =RENAMP
          0
1513 081BA 856  =RENAME ST=1   sRENAM          Set RENAME flag
1514 081BD 6D00         GOTO   COPY
1515                *
1516 081C1 0000         REL(5) =COPYDC         COPY Decompile
          0
1517 081C6 0000         REL(5) =COPYP          COPY Parse
          0
1518 081CB 21   =COPY   P=     1               Set no Leeway Check in allocation
1519 081CD 8E00         GOSUBL =ALINFO         Allocate Save area
          00
1520 081D3 445          GOC    CPYERX          Insufficient Memory
1521                *
```

```
1522            * Decipher File Information
1523            *   Clear Device information
1524            *   Blank fill chars 9-10 of Filename in R0
1525            *   Read next token
1526            *
1527 081D6 AF3   CPY010 D=0    W               Clear Device Info, D(S) must be 0,
1528 081D9 3302         LCASC  \  \
           02
1529 081DF 108          R0=C                   Blank file Chars 9-10 of filename
1530 081E2 14A          A=DAT0 B               Read next token
1531            *
1532            * COPY TO or COPY <spec> TO ...
1533            *
1534            * If Destination
1535            *    File spec or Keyword must follow
1536            *
1537 081E5 3100         LC(2)  =tTO
1538 081E9 966          ?A#C   B               Not TO ?
1539 081EC 31           GOYES  CPY030
1540 081EE 161          D0=D0+ 2               Skip TO token
1541 081F1 873          ?ST=1  sDEST           Destination ?
1542 081F4 C1           GOYES  CPY040          Check for Keyword or File spec
1543
1544            *
1545            * COPY TO
1546            * No source filename specified
1547            *    Get current filename and device
1548            *    go Shift device Info to D(A) & update info
1549            *
1550 081F6 8E00  CPY020 GOSUBL =DEFFIL         Get current filename/device
           00
1551 081FC 534          GONC   CPY065          B.E.T. go Save file info
1552            *
1553            * COPY w/o parameters or COPY <spec> EOL
1554            *   If Destination
1555            *      Filename <-- 0
1556            *      Device   =   0  (Mainframe)
1557            *   else
1558            *      Filename <-- Current filename
1559            *      Device   <-- Current file's device
1560            *
1561 081FF 310F  CPY030 LCHEX  F0              EOL terminator boundary
1562 08203 9E2          ?A<C   B               not EOL ?
1563 08206 A0           GOYES  CPY040
1564 08208 873          ?ST=1  sDEST           Destination ?
1565 0820B 41           GOYES  CPY050          Filename <-- 0
1566 0820D 58E          GONC   CPY020          Get current filename/device
1567            *
1568            * COPY ... CARD
1569            *    Filename = 0
1570            *    Device   = CARD
1571            *
1572 08210 3100  CPY040 LC(2)  =tCARD          CARD token
1573 08214 966          ?A#C   B
1574 08217 71           GOYES  CPY060
```

```
1575 08219 307          LC(1)  dCARD
1576 0821C A87          D=C    P              Save Device type
1577 0821F 161   CPY050 DO=DO+ 2              Skip token
1578 08222 AF0          A=0    W              No filename specified
1579 08225 5D1          GONC   CPY070         B.E.T.
1580               *
1581               * Error Exit
1582               *   Must take general BASIC Error Exit due to Polling
1583               *
1584 08228 8C00  CPYERX GOLONG =BSERR          Error Exit
           00
1585               *
1586               * Check for KEYS token
1587               *    tKYSck returns with:
1588               *       A = "keys"
1589               *       D(S) = 0    (MAIN)
1590               *       Token skipped
1591               *
1592 0822E 8E00  CPY060 GOSUBL =tKYSck        Check for KEYS token
           00
1593 08234 5B0          GONC   CPY065         KEYS found
1594
1595               *
1596               * No special keyword
1597               *    Execute File Specification
1598               *    If illegal --> Error Exit
1599               *    Shift device information to D(A)
1600               *
1601 08237 8E00         GOSUBL =FSPECx        Execute file specification
           00
1602 0823D 4AE          GOC    CPYERX         Illegal file spec
1603 08240 813   CPY065 DSLC                  Shift devicei info to D(A)
1604               *
1605               * Save File Information
1606               * If source
1607               *    Set destination flag
1608               *    go Decipher destination file information
1609               *
1610 08243 7312  CPY070 GOSUB  SVINFO
1611 08247 873          ?ST=1  sDEST          Destination ?
1612 0824A 90           GOYES  CPY080
1613 0824C 853          ST=1   sDEST          Set destination flag
1614 0824F 668F         GOTO   CPY010         go deciphter Destination file info
1615               *
1616               * DONE processing file names:
1617               *
1618               * If RENAME
1619               *    go execute RENAME
1620               * else
1621               *    Copy file
1622               *    If carry set ---> Error Exit
1623               *       else         ---> Return to Run Loop
1624               *
1625               *
1626 08253 876   CPY080 ?ST=1  sRENAM         RENAME?
```

```
1627 08256 D0          GOYES  renam
1628 08258 7D00        GOSUB  COPYu
1629 0825C 4BC         GOC    CPYERX          Error Return
1630 0825F 6000  Nxtstm GOTO   =NXTSTM        Return to Run Loop
1631              *
1632 08263 8C00  renam  GOLONG =RNM010
          00
```

```
1633                STITLE COPY Utility Routine
1634           ********************************************************************
1635           ********************************************************************
1636           **
1637           ** Name:(S) COPYu   -  COPY Utility
1638           **
1639           ** Category:   EXCUTL
1640           **
1641           ** Purpose:
1642           **      COPY Utility
1643           **      COPY Mainframe/PORTs
1644           **      COPY CARD
1645           **      COPY External
1646           **
1647           ** Entry:
1648           **      File information in SAVSTK area
1649           **      (Through SVINFO utility)
1650           **
1651           **      SAVSTK- 5 = Source Device Information
1652           **      SAVSTK-25 = Source Filename
1653           **      SAVSTK-30 = Destination Device Information
1654           **      SAVSTK-50 = Destination Filename
1655           **
1656           **      See SVINFO utility
1657           **          Device Info - Nib 0 = Device type
1658           **                        Nib 1-4 = Device specific info
1659           **          Filename      Up to 10 chars
1660           **                        Blank filled
1661           **
1662           ** Exit:
1663           **      Save area is RELEASED
1664           **
1665           **      Carry clear - Good COPY
1666           **        R1 = Start of file just copied
1667           **             If destination into Mainframe/IRAM
1668           **
1669           **      pCOPYx Poll issued if either Source or Destination
1670           **             is External (not Mainframe/IRAM/CARD)
1671           **
1672           **      Carry set - Error Return
1673           **        C(4) = Error number
1674           **
1675           **        Error Returns:
1676           **
1677           **        eMEM      - No memory to create destination file
1678           **        eFSPEC    - No response to COPY Poll
1679           **                    Non Mainframe for CARD
1680           **        eFnFND    - Source file not found
1681           **        eFPROT    - Private source file
1682           **        eFEXST    - Destination file exists
1683           **        eFTYPE    - Non KEYS file for KEYS copy
1684           **        eFACCS    - Destination is unknown PORT device
1685           **        eDVCNF    - PORT device not found
1686           **
1687           ** Calls:      FILFIL,POLL,FINDF (FINDFS),GETPR1,MFDVC+,RDHDR1,
```

```
1688            **            CRETF+,MOVEUO,RDINFD,WFTMD-,LEXBF+,RLINFO,CRDFIL,
1689            **            FILCRD,CHAIN-,BASCHA,FLDEV+,MFDEVC,D1=SRO
1690            **
1691            ** Uses.......
1692            **  Exclusive: A-D,D0,D1,SAVSTK (50 nibs),R0,R1,R2,
1693            **             S0-S7,S8,S9,S12,STMTR0 (5 nibs),SCRTCH (32 nibs)
1694            **             4 levels of RSTKBF (if Copying LEX file)
1695            **                                 See LEXBF+
1696            **
1697            **  Inclusive:
1698            **
1699            **      sDEST  = Destination Execute flag  (S3)
1700            **      sREADI = Read file information     (S4)
1701            **      sKEYS  = COPY to KEYS              (S5)
1702            **      sPCRD  = Private CARD              (S8)
1703            **          D  = Device information
1704            **                 D(0) = Device Type
1705            **                        F = No device
1706            **                 dMAIN  0 = :MAIN
1707            **                 dPORT  1 = :PORT
1708            **                            D(1,2) = Extender#, Port#
1709            **                                   = FF if all ports
1710            **                 dCARD  7 = CARD     D(B)=0
1711            **                 dPCRD  7 = :PCRD    D(B)#0
1712            **                     >= 8 = PIL / Non-mainframe Device
1713            **
1714            **        R1  = Destination file start    (CREATF)
1715            **        R3  = Start of source file
1716            **
1717            **
1718            **           POLL   uses B,C,AVMEME,XM
1719            **           FINDF  use  A-D,D0,D1,S6,S8,S9,R2,R3
1720            **           MOVEUO uses A,C,D0,D1,P
1721            **           RDINFO
1722            **           SVINFO use  A,C,D,S3,S4
1723            **           GETPR1 uses A,C,D1
1724            **           CREATF uses A-D,D0,D1,R0,R1,SCRTCH (32 nibs)
1725            **           tKYSck uses A,C,D(S)
1726            **           LEXBF+ uses A-D,R1,4 lvls RSTKBF
1727            **
1728            ** Stk lvls:   6
1729            **
1730            ** Algorithm:
1731            **
1732            ** COPYu:
1733            **      Get source information, Fill missing names (FILFIL)
1734            **      If device # Mainframe        (sEXTDV) or ext PORT
1735            **  0:     POLL for COPY external device      (pCOPYx)
1736            **         If carry set
1737            **            Error Return
1738            **         If no response
1739            **            If external device              (D(0)>7)
1740            **               Error <-- eFSPEC
1741            **            If unknown PORT device
1742            **               Error <-- eFACCS
```

```
1743          **          else
1744          **            RTN
1745          **
1746          **  1:  If MAIN | PORT              (sCARD=0)
1747          **          If source
1748          **             Find source file                  (FINDF)
1749          **             Save pointer to file start        (R3)
1750          **             Check file protection             (GETPR1)
1751          **               Error Return if private         (eFPROT)
1752          **             If BASIC file                     (BASCHA)
1753          **                Chain file                     (CHAIN-)
1754          **             Set Destination flag
1755          **             Get Destination device info       (MFDVC+)
1756          **             If CARD
1757          **                go COPY to CARD                (goto 5)
1758          **          else   (destination)
1759          **             If "keys" filename
1760          **                 Set KEYS File flag
1761          **             Save source start                 (STMTR0)
1762          **             If PORT destination not found     (FLDEV+)
1763          **                Error                          (eDVCNF)
1764          **             If Not Mainframe destination
1765          **                Convert Dest. filename to Uppercase (CVUCW)
1766          **                Save updated Dest. File infor (SVINF+)
1767          **             If not Independent RAM or MAIN
1768          **                go Poll for COPY to unknown dev (goto 0)
1769          **             Find destination file             (FINDF)
1770          **             If file found
1771          **                Error Return                   (eFEXST)
1772          **             Restore source start
1773          **             Read Source file header           (RDHDR1)
1774          **             If KEYS Copy                      (sKEYS)
1775          **                If source file type # KEYS
1776          **                   Error Return
1777          **             Compute file length
1778          **             Create Destination file           (CRETF+)
1779          **               Error if not created          (Carry set)
1780          **             Copy source to destination        (MOVEU3)
1781          **             Read destination information      (RDINFD)
1782          **             Write new filename
1783          **             Write new creation date & time
1784          **      4:     If LEX file copy   (Dest. filetype = LEX)
1785          **                Save file start (R1 --> RSTK)
1786          **                Regenerate LEX Buffer          (LEXBF+)
1787          **                Restore file start             (R1)
1788          **                goto Done;
1789          **
1790          **          If CARD | PCRD device
1791          **             If source
1792          **                Set destination flag
1793          **                Read destination device          (MFDVC+)
1794          **                R3 <-- Source Filename
1795          **                R2 <-- Destination filename
1796          **                If destination device = MAIN
1797          **                   Copy CARD to File             (CRDFIL)
```

```
1798            **              Set R1 = Last file in Mainframe(EOFLCH)
1799            **              Position to File type
1800            **              go Check if Lex File copy      (goto 4)
1801            **            else
1802            **              Error Exit                     (eFSPEC)
1803            **    5:    If destination = CARD
1804            **            If Private Card                ((D(1-2)#0)
1805            **              Set Private Card flag
1806            **            If source device = MAIN | PORT
1807            **              R1 <-- Destination Filename
1808            **               C <-- Source file start
1809            **              Copy file to CARD              (FILCRD)
1810            **            else
1811            **              Error Exit                     (eFSPEC)
1812            **
1813            ** Done:  Release File Informatin Save area     (SRLEAS)
1814            **        Return CC
1815            **
1816            ** CPYERR: Save error message on stack
1817            **         Release File information save area
1818            **         Restore error message
1819            **         Return SC
1820            **
1821            ** History:
1822            **
1823            **    Date     Programmer          Modification
1824            **  --------   ----------  ----------------------------------
1825            **  07/04/82   JP          Modified documentation
1826            **  11/20/82   JP          Fixed COPY TO CARD
1827            **  12/18/82   JP          Combined pCOPYd with pCOPYx
1828            **  12/18/82   JP          Added chain source if BASIC
1829            **  03/21/83   JP          Test if PORT not found after FLDEV+
1830            **  03/21/83   JP          Using S-R0-0 to save Source start
1831            **  05/11/83   JP          Packed CVUCW,SVINF+ calls @ CPY135
1832            **
1833            ********************************************************************
1834            ********************************************************************
1835            *
1836            ********************************************************************
1837            ********************************************************************
1838            **
1839            ** Name:(S) pCOPYx  -  Poll for COPY to external device
1840            **
1841            ** Category:   POLL
1842            **
1843            ** Type:       POLL
1844            **
1845            ** Purpose:
1846            **      Poll for COPY utility execute
1847            **      External source or destination file specifier found OR
1848            **      Destination device on PORT is of unknown type
1849            **
1850            ** Should poll be "Handled" (return with XM=0)?:
1851            **      Yes - If successful COPY occurs
1852            **
```

```
1853            ** Meaning of "Handling" Poll (what does code do if handled?):
1854            **      COPY source file to destination file on appropriate
1855            **      device
1856            **
1857            ** Entry conditions for handler (registers, ST, RAM, etc.):
1858            **      B[A] = Poll number (pCOPYx)
1859            **      HEX mode.
1860            **      P=0.
1861            **
1862            **      If D(0) = External Device (D(0)>=8)
1863            **         sEXTDV = 1 (S0)
1864            **         sUNDEF = 1 (S1) if both filenames undefined = 0
1865            **         sDEST  = 0 (S3)
1866            **         A      = First 8 characters of source filename
1867            **                  Blanked filled
1868            **         R0(0-3)= Last 2 characters of source filename
1869            **                  Blanked filled if none
1870            **         D(A)   = Source device information from RDINFO
1871            **                  D(0)  = Device type
1872            **                  D(1-4)= Devices internal coding
1873            **
1874            **                  HPIL used Device 8 --- see NOTE below
1875            **         R2     = Destination device info from RDINFO
1876            **         SAVSTK   holds source and destination information
1877            **                  (See Special Memory/Pointer Considerations)
1878            **
1879            **      If D(0) = Unknown device type (1 < D(0) < 7)
1880            **         A      = First 8 characters of destination filename
1881            **                  (blank filled)
1882            **         R0(0-3)= Last two characters of destination filename
1883            **                  (blank filled)
1884            **         D(0)   = Device type
1885            **         D(1)   = Extender#
1886            **         D(2)   = PORT #
1887            **         STMTRO = Start of source file
1888            **         SAVSTK   holds source and destination information
1889            **                  (See Special Memory/Pointer Considerations)
1890            **
1891            ** Normal exit conditions from handler if handled (ST, RAM,
1892            ** registers, etc.):
1893            **      Carry clear
1894            **      HEX mode.
1895            **      XM=0.
1896            **      R1 = Start of file just copied if TO MAINFRAME
1897            **      Source file copied to destination file on appropriate
1898            **      device.
1899            **
1900            ** Normal exit conditions from handler if not handled (ST, RAM,
1901            ** registers, etc.):
1902            **      Carry clear
1903            **      HEX mode.
1904            **      XM=1.
1905            **
1906            ** Error exit conditions from handler:
1907            **      Carry set.
```

```
1908            **      HEX mode.
1909            **      C[0-3] = Error number.
1910            **      COPY was not sucessful due to indicated Error Number
1911            **
1912            ** Available subroutine levels: 6
1913            **      POLL handler is one level shallower than caller--
1914            **      COPYu uses 6 levels; The handler must be able to
1915            **      Return to POLL
1916            **
1917            ** NOTE:
1918            **      HPIL uses Device Type=8
1919            **        This device type is set in response to pFSPCx poll
1920            **        when the file specifier is being evaluated
1921            **      Other device handlers must be assigned their special
1922            **        special device type by the Resource Allocation Czar
1923            **        (See HP-71 IMS Volume 1)
1924            **      Respond to pFILXQ for non HPIL device to gain control
1925            **        of the File Specification execute
1926            **
1927            **      Devices on PORTS (ex: EEPROM) should use Device types
1928            **      between 3 and 6.  This device type will be encoded in
1929            **      the ID of the module plugged into a PORT.
1930            **        These Device types must be assigned by the Resource
1931            **        Allocation Czar (see HP-71 IMS Volume 1)
1932            **
1933            ** What registers/RAM may be used if handled?:
1934            **      A-D, D0, D1, P
1935            **      R0,R1,R2,S2,S3,S4,S5,S6,S7,S8,S9
1936            **      Dont' use STMTD0 (saved status for CHAIN)
1937            **
1938            ** What registers/RAM may be used if not handled?:
1939            **      A-D, D0, D1, P
1940            **      R1,S4,S5,S6,S7,S8,S9
1941            **      Don't use STMTD0 (saved status for CHAIN)
1942            **
1943            ** What registers/RAM may be used if error exit (POLL only)?:
1944            **      A-D, D0, D1, P
1945            **      R0,R1,R2,S2,S3,S4,S5,S6,S7,S8,S9
1946            **      Don't use STMTD0 (saved status for CHAIN)
1947            **
1948            ** Special memory/pointer considerations (are pointers funny?):
1949            **
1950            **      The SAVSTK area has been moved toward LOW memory due to
1951            **      the issuing of the POLL.  Therefore, all offsets into
1952            **      the SAVSTK area must SUBTRACT the lPOLSV (62 decimal)
1953            **      from the SAVSTK pointer to access the file information.
1954            **
1955            **      Saved information:
1956            **      SAVSTK-5 (- lPOLSV) = Source Device information  5 nibs
1957            **      SAVSTK-25(- lPOLSV) = Source filename           20
1958            **      SAVSTK-30(- lPOLSV) = Destination Device info    5
1959            **      SAVSTK-50(- lPOLSV) = Destination filename      20
1960            **
1961            ** Envisioned application(s):
1962            **
```

```
1963            **      Allow COPY TO filename:TAPE
1964            **      Allow COPY TO filename:PORT(1) where EEPROM in PORT(1)
1965            **      Allow COPY TO a special device in a PORT
1966            **      Allow COPY TO an external device NOT HPIL
1967            **
1968            ** History:
1969            **
1970            **    Date     Programmer              Modification
1971            **   --------  ----------  ------------------------------------
1972            **  07/19/82   JP          Added documentation
1973            **  12/18/82   JP          Combined pCOPYd with pCOPYx
1974            **  03/21/83   JP          Changed entry conditions (STMTRO)
1975            **  05/11/83   JP          Modified documentation
1976            **  08/11/83   JP          Restricted STMTDO usage
1977            **
1978            ************************************************************
1979            ************************************************************
1980            *
1981            * COPY Utility Entry Point:
1982            *
1983 08269      =COPYu
1984            *
1985            * Get Source information; Fill missing file names
1986            *
1987 08269 8E00         GOSUBL =FILFIL        Get source info; Fill filenames
         00
1988 0826F 860          ?ST=0  sEXTDV         Not External Device ?
1989 08272 C4           GOYES  CPY100
1990            *
1991            * If non-Mainframe device (sEXTDV) or unknown PORT device
1992            *    POLL for COPY
1993            *    If carry set
1994            *       Error return
1995            *    If response
1996            *       Return
1997            *    If external device  (D(0) >=8)
1998            *       Error <-- eFSPEC
1999            *    else        (unknown port device)
2000            *       Release COPY save space
2001            *       Error <-- eFACCS
2002            *
2003 08274 79D1  CPY095 GOSUB  Poll
2004 08278 80           CON(2) =pCOPYx
2005 0827A 4D1          GOC    CPYER1         Error return from POLL
2006 0827D 831          ?XM=0                 Response to POLL ?
2007 08280 C1           GOYES  CPY098         Yes
2008 08282 A07          D=D+D  P              External device ?
2009 08285 4A0          GOC    CPYER0
2010 08288 7571         GOSUB  CPYEXT         Release save area
2011 0828C 6343         GOTO   CRTI60         "eFACCS" Error Return
2012 08290 20    CPYER0 P=     0              No response to COPY
2013 08292 3300         LC(4)  =eFSPEC        Illegal File Spec
         00
2014 08298 6E61  CPYER1 GOTO   CPYERR         Error Return
2015 0829C 6461  CPY098 GOTO   CPYEXT         Exit with successful copy
```

```
2016                *
2017                * Jump Back point:
2018                *
2019                * COPY File to CARD
2020                *    R1 <-- Destination Filename
2021                *    C <-- Source start
2022                *    If Destination Device = PCRD  [ D(1-2)#0 ]
2023                *       Set Private Card flag
2024                *    Copy file to CARD
2025                *    Restore stack levels
2026                *
2027 082A0 101   CPY2CD R1=A                  Destination filename
2028 082A3 11B         C=R3                  Source start
2029 082A6 848         ST=0   sPCRD          Clear Private Card flag
2030 082A9 F7          DSR    A              Shift off Device type
2031 082AB 96B         ?D=0   B              Not private card ?
2032 082AE 50          GOYES  CPY250
2033 082B0 858         ST=1   sPCRD          Set Private Card flag
2034 082B3 8F00  CPY250 GOSBVL =FILCRD        Copy file to CARD
           000
2035 082BA 6641        GOTO   CPYEXT         Clean-up and return
2036                *
2037                * If CARD device
2038                *    If Source = CARD
2039                *       go Copy CARD
2040                * If MAIN | PORT source device
2041                *    Find source file
2042                *
2043 082BE 862   CPY100 ?ST=0  sCARD          not CARD device ?
2044 082C1 11          GOYES  CPY105         No Device, :MAIN, :PORT
2045 082C3 8E00        GOSUBL =MFDEVC        Check Device type of source
           00
2046 082C9 862         ?ST=0  sCARD          Source is not CARD device ?
2047 082CC 60          GOYES  CPY105         go Find File
2048 082CE 6241        GOTO   CPYCRD         COPY CARD
2049 082D2 7271  CPY105 GOSUB  FINDFS         Shift Device type to D(S),Find Sour
2050 082D6 41C         GOC    CPYER1
2051                *
2052                * Save start of source file
2053                * Check Privacy on source file
2054                *    Exit from GETPRT:
2055                *      D1 @ File type field of Source file header
2056                * If source file type = BASIC
2057                *    Chain file before copying
2058                *
2059 082D9 137         CD1EX
2060 082DC 10B         R3=C                  Save start of source file
2061 082DF 137         CD1EX
2062 082E2 8E00        GOSUBL =GETPR1        Check protection
           00
2063 082E8 4FA         GOC    CPYER1         "File protection" error
2064 082EB D0          A=0    A
2065 082ED 15B3        A=DAT1 4              Read file type
2066 082F1 8E00        GOSUBL =BASCHA        Check if BASIC file type
           00
```

```
2067 082F7 490          GOC    CPY110          Not BASIC
2068 082FA 113          A=R3                   Source file start
2069 082FD 7000         GOSUB  =CHAIN-         Chain BASIC file
2070               *
2071               * Done with Source Execute
2072               *    Set Destination flag
2073               *    Get Destination Information
2074               *
2075 08301 853   CPY110 ST=1   sDEST
2076 08304 8E00         GOSUBL =MFDVC+          Get dest information
           00
2077 0830A 872          ?ST=1  sCARD           CARD device ?
2078 0830D 39           GOYES  CPY2CD          COPY TO CARD
2079               *
2080               * MAIN | PORT Device
2081               *   Destination Execute     D(0)=Dest. Device; C(0)=dCARD
2082               *     If destination filename = "keys"
2083               *        Set KEYS Copy flag
2084               *
2085 0830F 845   CPY120 ST=0   sKEYS           Clear KEYS Copy flag
2086 08312 3FB6         LCASC  \    syek\      "keys" Filename
           5697
           3702
           0202
           02
2087 08324 976          ?A#C   W              Not "keys" destination ?
2088 08327 50           GOYES  CPY130
2089 08329 855          ST=1   sKEYS           Set KEYS Copy flag
2090               *
2091               * Save Source file start in STMTRO (since R3 gets used)
2092               * Map Device code into specific code
2093               * If Carry set
2094               *    If PORT not found   D(A)=0
2095               *       Error ---> Device not Found
2096               * If not IRAM
2097               *    Poll to COPY to unknown device
2098               *
2099 0832C 11B   CPY130 C=R3                   Save Source File start (R3 gets use
2100 0832F 7A4C         GOSUB  D1=sRO          in STMTRO
2101 08333 145          DAT1=C A
2102 08336 8E00         GOSUBL =FLDEV+         Shift device, map device code
           00
2103 0833C 570          GONC   CPY135          Device okay
2104 0833F 8AB          ?D=0   A               PORT not found ?
2105 08342 32           GOYES  CPYER2          Error --> Device not Found
2106 08344 813   CPY135 DSLC                   Move device info to D(A)
2107 08347 8AB          ?D=0   A               Mainframe device ?
2108 0834A E0           GOYES  CPY140
2109 0834C 301          LC(1)  =dIRAM          Independent RAM Device type
2110 0834F 903          ?D=C   P               Independent RAM ?
2111 08352 60           GOYES  CPY140
2112 08354 6F1F         GOTO   CPY095          Poll for COPY to unknown device
2113               *
2114               * Move Device type to D(S) for FIND File
2115               * Try to find Destination file across specified Device
```

```
2116                *
2117 08358 7CE0    CPY140 GOSUB  FINDFS        Shift Device type, Find file
2118 0835C 4C0            GOC    CPY160        File not found
2119                *
2120                * Destination file found
2121                *   Error exit
2122                *
2123 0835F 3300           LC(4)  =eFEXST       "File exists"
           00
2124 08365 61A0    CPYER2 GOTO   CPYERR
2125                *
2126                * Retrieve Destination Device Info and shift it for CRETF+
2127                * Restore Source file start
2128                * Read source file header
2129                * Transfer Source file start to R2; File length to R0
2130                * If KEYS Copy
2131                *    If source file type # KEYS
2132                *       Error exit
2133                *
2134 08369 8E00    CPY160 GOSUBL =MFDVC+         Get Destination info
           00
2135 0836F 817            DSRC                 Shift device infor for CRETF+
2136 08372 770C           GOSUB  D1=sR0
2137 08376 147            C=DAT1 A             Restore Source file start
2138 08379 108            R0=C                 Save it back into R0
2139 0837C 135            D1=C
2140 0837F 8E00           GOSUBL =RDHDR1       Read file header
           00
2141 08385 120            AR0EX                R0 <-- File length; A <-- File star
2142 08388 122            AR2EX                R2 <-- File start;  A <-- File type
2143 0838B 865            ?ST=0  sKEYS         not KEYS Copy ?
2144 0838E 71             GOYES  CPY170
2145 08390 34C0           LC(5)  =fKEY
           2E0
2146 08397 8A2            ?A=C   A             KEY file type ?
2147 0839A B0             GOYES  CPY170
2148 0839C 3300           LC(4)  =eFTYPE       "Illegal file type"
           00
2149 083A2 52C            GONC   CPYER2        B.E.T.
2150                *
2151                * Compute File length
2152                * Attempt to Create file
2153                *
2154 083A5 110    CPY170 A=R0                  A <-- File length
2155 083A8 D2             C=0    A
2156 083AA 3102           LC(2)  =oFLENh       Offset to file length in header
2157 083AE C2             C=C+A  A             Total file length
2158 083B0 7011           GOSUB  CRETF+        Create file in RAM / IRAM
2159 083B4 40B            GOC    CPYER2        Error Return from Create File
2160                *
2161                * Copy Source to Destination
2162                *    D0 = Start of Source
2163                *    D1 = Start of Destination
2164                *     B = Length to Move
2165                *
```

```
2166 083B7 11A   CPY177 C=R2                  Source start
2167 083BA 134          DO=C
2168 083BD 119          C=R1                  Destination start (from CREATF)
2169 083C0 135          D1=C
2170 083C3 8F00         GOSBVL =MOVEUO        Copy source to destination
           000
2171               *
2172               * Read destination filename
2173               * Write destination filename to created file
2174               * Update date and time
2175               *
2176 083CA 7A90         GOSUB  RDINFD
2177 083CE 119          C=R1                  Destination start
2178 083D1 134          DO=C
2179 083D4 1507         DATO=A W              Write destination filename
2180 083D8 7AF1         GOSUB  =WFTMD-        Write time,date; don't zero COPY ni
2181               *
2182               * If LEX file Copy
2183               *    Update Lex Buffer Information
2184               *
2185 083DC 186          DO=DO- (oTIMEh)-(oFTYPh)+1 Pos to File type-1
2186 083DF 142   CPY180 A=DATO A              Read File type @ File type-1
2187 083E2 F4           ASR    A              Shift file type to A(0-4)
2188 083E4 3480         LC(5)  =fLEX          LEX file type
           2E0
2189 083EB 8A6          ?A#C   A              Not LEX file copy ?
2190 083EE 31           GOYES  CPYEXT
2191 083F0 119          C=R1                  Save Start of file
2192 083F3 06           RSTK=C
2193 083F5 8F00         GOSBVL =LEXBF+        Update LEX Buffer infor
           000
2194 083FC 07           C=RSTK                Restore start of file
2195 083FE 109          R1=C
2196               *
2197               * CPYEXT: Successful Copy
2198               *    Clear CARD Copy flag
2199               *    Release File Information Save area
2200               *    Return CC
2201               *
2202 08401 8C00  CPYEXT GOLONG =RLINFO         Release save area
           00
2203               *
2204               * CPYERR - Error Return
2205               *
2206 08407 06    =CPYERR RSTK=C                Save Error number
2207 08409 74FF         GOSUB  CPYEXT         Release save area
2208 0840D 07           C=RSTK
2209 0840F 02           RTNSC                 Return SC
2210               *
2211               * COPY CARD to File
2212               *      R3 <-- Source filename
2213               *      Read destination information
2214               *      R2 <-- Destination filename
2215               *
2216 08411 103   CPYCRD R3=A                   Source filename
```

```
2217 08414 853          ST=1   sDEST          Set Destination flag
2218 08417 8E00         GOSUBL =MFDVC+        Get Dest. information
           00
2219 0841D 102          R2=A                  Destination filename
2220               *
2221               * Check Destination device
2222               * If CARD | PORT
2223               *    Error Exit
2224               * else
2225               *    Copy CARD to file
2226               *
2227 08420 862          ?ST=0  sCARD          not CARD device ?
2228 08423 60           GOYES  CPY220         (External devices have been elimina
2229 08425 6A6E  CPYER3 GOTO   CPYERO         Illegal File Spec
2230 08429 301   CPY220 LC(1)  dPORT          PORT Device
2231 0842C 903          ?D=C   P              PORT Device ?
2232 0842F 6F           GOYES  CPYER3         Error exit
2233 08431 8F00         GOSBVL =CRDFIL        Copy CARD to file
           000
2234               *
2235               * R2 @ Filestart
2236               * Set R1 @ start of file just copied
2237               *    Find end of file chain
2238               * Position to File type
2239               * go Check if Lex File COPY
2240               *
2241 08438 11A          C=R2
2242 0843B 109          R1=C                  R1 @ start of file just copied
2243 0843E 134          D0=C
2244 08441 16E          D0=D0+ (oFTYPh)-1     Position to File type - 1
2245 08444 6A9F         GOTO   CPY180         go Check if Lex File copy
2246               *
2247               *
2248 08448 817  =FINDFS DSRC                  Move Device type to D(S)
2249 0844B 8C00         GOLONG =FINDF         Find file
           00
2250 08451 8C00  Poll   GOLONG =Polly
           00
```

```
2251                    $TITLE Save and Read File Information
2252            ********************************************************************
2253            ********************************************************************
2254            **
2255            ** Name:(S) SVINFO  -  Save/Read File Information
2256            ** Name:(S) SVINF+  -  Save/Read File Information
2257            ** Name:(S) RDINFO  -  Read Source/Dest File Information
2258            ** Name:    RDINFS  -  Read Source File Info
2259            ** Name:    RDINFD  -  Read Dest File Info
2260            **
2261            ** Category:   SAVSTK
2262            **
2263            ** Purpose:
2264            **      These entry points are used by COPY, TRANSFORM,
2265            **      RUN, and CHAIN to save and access information on
2266            **      their source/destination files.  The info is
2267            **      stored in an area on the SAVSTK, which must be
2268            **      allocated using ALINFO beforehand.  SVINFO and
2269            **      SVINF+ write the file info; RDINFO, RDINFS, and
2270            **      RDINFD read the info back.
2271            **
2272            ** Entry:
2273            **      All:    File Info save area allocated on SAVSTK
2274            **
2275            **      SVINFO:
2276            **       A      =  Filename   (first 8 chars)
2277            **       R0(3-0)=  Last 2 chars of filename
2278            **       D(A)   =  Device information
2279            **         D(0)   =  Device code
2280            **         D(4-1) =  Device spec (Port, extender#, etc)
2281            **                   If PORT:
2282            **                      D(1) = Extender#
2283            **                      D(2) = PORT#
2284            **                   If HPIL:
2285            **                      D(3-1) = Device address
2286            **                      D(4)   = Device characterization
2287            **       S3     =  0 => Save info in source file position
2288            **              =  1 => Save info in dest file position
2289            **
2290            **      SVINF+:  Same as SVINFO, except:
2291            **       D(S)   = Device code (position returned by FSPECx)
2292            **       D(3-0) = Device spec shifted right (in position
2293            **                  returned by FPECx)
2294            **
2295            **      RDINFO:
2296            **       S3     =  0  if Source file info to be read
2297            **              =  1  if Dest file info to be read
2298            **
2299            **      RDINFS, RDINFD:
2300            **       None.
2301            **
2302            ** Exit:
2303            **       S4     =  0  (SVINFO, SVINF+)
2304            **              =  1  (RDINFO, RDINFS, RDINFD)
2305            **       S3     =  0  (RDINFS)
```

```
2306            **            =  1  (RDINFD)
2307            **            =  Entry condition (RDINFO)
2308            **
2309            **     SVINFO, SVINF+:  Information saved in appropriate spot
2310            **      A      =  Entry Condition
2311            **      R0(3-0)=  Entry Condition
2312            **      D(A)   =  Device information (see SVINFO entry)
2313            **
2314            **     RDINFO:  Info on selected file
2315            **      A      =  Filename  (first 8 chars)
2316            **      R0     =  Last 2 chars of filename
2317            **      D(A)   =  Device information (see SVINFO entry)
2318            **      C(A)   =  D(A)
2319            **
2320            **      RDINFS:  Same as RDINFO; Source information
2321            **      RDINFD:  Same as RDINFO; Destination information
2322            **
2323            ** Calls:      None.
2324            **
2325            ** Uses.......
2326            **  Inclusive: sDEST(S3),sREADI(S4),A,C,R0,D1,
2327            **             D(A) (RDINFO, RDINFD, RDINFS),
2328            **             D    (SVINF+)
2329            **
2330            ** Stk lvls:   0
2331            **
2332            ** Detail:
2333            **      Start addr  Size    Information
2334            **      ----------  ----  ------------------------------
2335            **      SAVSTK-50    20   Destination Filename
2336            **      SAVSTK-30     5   Destination Device Information
2337            **      SAVSTK-25    20   Source Filename
2338            **      SAVSTK- 5     5   Source Device Information
2339            **
2340            ** History:
2341            **
2342            **    Date     Programmer             Modification
2343            **  --------   ----------   ------------------------------------
2344            **  07/04/82   JP           Modified documentation
2345            **
2346            ************************************************************************
2347            ************************************************************************
2348 08457 813  =SVINF+ DSLC                  Shift Device type to D(0)
2349 0845A 844  =SVINFO ST=0   sREADI         Clear READ info flag
2350 0845D 6010         GOTO   RDIN10
2351 08461 843  =RDINFS ST=0   sDEST          Clear Destination flag
2352 08464 6600         GOTO   RDINFO
2353 08468 853  =RDINFD ST=1   sDEST          Set Destination flag
2354 0846B 854  =RDINFO ST=1   sREADI         Set READ infor flag
2355 0846E 1FE9  RDIN10 D1=(5) =SAVSTK
           5F2
2356 08475 147          C=DAT1 A              Start of save area
2357 08478 135          D1=C
2358            *
2359            * Position to source information
```

```
2360                *
2361 0847B 1C8        D1=D1- (lDEVC)+(lFNAM+)  Skip Device,Last 2 chars
2362 0847E 1CF        D1=D1- lFNAM8            Skip to front of filename
2363                *
2364                * If Destination --> Position to destination information
2365                *
2366 08481 863        ?ST=0  sDEST             Source information ?
2367 08484 80         GOYES  RDIN20
2368 08486 1C8        D1=D1- (lDEVC)+(lFNAM+)  Skip Device,Last 2 chars
2369 08489 1CF        D1=D1- lFNAM8            Skip to front of filename
2370 0848C 874  RDIN20 ?ST=1  sREADI           Read information ?
2371 0848F A1         GOYES  RDIN30
2372                *
2373                * Save information
2374                *
2375 08491 1517       DAT1=A W                 Save first 8 chars of filename
2376 08495 17F        D1=D1+ lFNAM8            Skip over first 8 chars
2377 08498 118        C=R0
2378 0849B 15D3       DAT1=C 4                 Save last 2 chars of filename
2379 0849F 173        D1=D1+ lFNAM+            Skip over last 2 chars
2380 084A2 DB         C=D    A
2381 084A4 145        DAT1=C A                 Save Device information
2382 084A7 03         RTNCC
2383                *
2384                * Read information
2385                *
2386 084A9 1537 RDIN30 A=DAT1 W                 Read first 8 chars of filename
2387 084AD 17F        D1=D1+ lFNAM8            Move over first 8 chars
2388 084B0 15F3       C=DAT1 4                 Read last 2 chars of filename
2389 084B4 108        R0=C
2390 084B7 173        D1=D1+ lFNAM+            Move over last 2 chars
2391 084BA 147        C=DAT1 A                 Read device information
2392 084BD D7         D=C    A
2393 084BF 03         RTNCC
```

```
2394                    STITLE Create Mainframe File
2395            ********************************************************************
2396            ********************************************************************
2397            **
2398            ** Name:    CREATF  -  Create File in MAIN
2399            ** Name:(S) CRETF+  -  Create file in MAIN or in IRAM
2400            **
2401            ** Category: FILUTL
2402            **
2403            ** Purpose:
2404            **      Create a file in designated RAM device.
2405            **
2406            ** Entry:
2407            **      CREATF:
2408            **       C(A) = Total memory size of new file in nibbles
2409            **              (must include length of file header)
2410            **      CRETF+:
2411            **       C(A) = Total memory size of new file in nibbles
2412            **              (must include length of file header)
2413            **       D(S) = 0 or F  => Create in mainframe
2414            **            = other   => Create in PORT
2415            **                 D(B) determines in which port to create:
2416            **                   D(1) = PORT #
2417            **                   D(0) = Extent #
2418            **                 D(B) = FF => Create on first avail. port
2419            **
2420            **
2421            ** Exit:      R1 @ Start of new file  (from WFTMDT)
2422            **            B(A) = Total memory size of new file
2423            **           CARRY SET => NEW FILE WAS NOT CREATED
2424            **            C(3-0)   =   Error number
2425            **           CARRY CLR => FILE CREATED SUCCESSFULLY
2426            **            The following header info filled in:
2427            **            Flag field and COPY code field zeroed
2428            **            Creation time and date
2429            **            File chain length
2430            **
2431            ** Calls:     MOVED3, RFADJ+, WFTMDT, EOFLC+, ROMF-1, WFLENG
2432            **            LSTADR, ROMCHK, ROMFND, MEMCKL, RCO1, RAMROM
2433            **
2434            ** Uses:      A-D,D0,D1,R0,R1,SCRTCH (32 nibs),S0-S7 (YMDHMS)
2435            **
2436            ** Detail:
2437            **            B = Size of new file          (Offset for pointers)
2438            **            R0= Size of new file     (Saved during WFTMDT call)
2439            **            R1 = Start of new file
2440            **
2441            **
2442            **
2443            ** Algorithm:
2444            **            Save size of new file (R0)
2445            **            If not Mainframe create              D(S) >= 1
2446            **               If PORT not specified             D(B) = FF
2447            **      1:          Find first avail port          (ROMCHK)
2448            **                    Error if no ports
```

```
2449            **                  Try to create file on port     (CRTPRT)
2450            **                  If not successful
2451            **                     Try next port               (goto 1)
2452            **               else
2453            **                  Find specified port            (ROMF-1)
2454            **                    Error if not found
2455            **  CRTPRT:         Error if Port not RAM          (RAMROM)
2456            **                  Calc end of file chain
2457            **                  Calc last address on Port      (LSTADR)
2458            **                  If enough memory
2459            **                     Write zero byte @ file chain end
2460            **                     Back up to file header
2461            **                     Write Date and time         (WFTMDT)
2462            **                     Write file length
2463            **             else
2464            **               Check if enough memory w/LEEWAY to create
2465            **               Read End of Source (AVMEMS) --->(D0)
2466            **               End of Destination AVMEMS + File size --> (D1)
2467            **               Length of Source = End of Source - Begin of Source
2468            **                                = AVMEMS - (MAINEN - 2)
2469            **               Begin Source @ Zero byte of File Chain
2470            **               Move memory down                (MOVED3)
2471            **               Zero flags,write Time, Date to hdr(WFTMDT)
2472            **               Write File length chain to header
2473            **               Save PRGMEN, CURREN             (R1)
2474            **               Adjust memory & stack pointers  (RFADJ+)
2475            **               Restore PRGMEN, CURREN
2476            **
2477            ** Stack lvls: 5
2478            **               4 if file created in MAIN
2479            **
2480            ** History:
2481            **
2482            **    Date      Programmer   Modification
2483            **  --------    ----------   ------------
2484            **  06/30/82    S.W.         Added documentation
2485            **  07/15/82    JP           Modified D(S) entry conditions
2486            **  10/11/82    JP           Added LEEWAY check for MemChk
2487            **  12/17/82    S.W.         Eliminated check for ROM -
2488            **                            Trapped out in poll, as with
2489            **                            other non-RAM memory devices
2490            **  01/10/83    S.W.         Eliminated poll to CREATE on
2491            **                            non-RAM device
2492            **  01/31/83    S.W.         Always uses 5 stack levels
2493            **  03/17/83    JP           Packed D1=(5) =MAINEN
2494            **  06/23/83    S.W.         When adding file to an IRAM, now
2495            **                            we guard against 'wrap-around'.
2496            **                            Replaced GOVLNG RMEM w/
2497            **                            GOLONG RMEM10.
2498            **  06/29/83    S.W.         Don't save CURREN on RSTK before
2499            **                            calling RFADJ+ - uses too many
2500            **                            levels - use R1 instead.
2501            **
2502            ********************************************************************
2503            ********************************************************************
```

```
2504                *
2505 084C1 AC3   =CREATF D=0    S              Force Mainframe create
2506 084C4 108   =CRETF+ RO=C   A              Save new file size
2507 084C7 ACB           C=D    S
2508 084CA B46           C=C+1  S
2509 084CD 4A0           GOC    CRETMF
2510 084D0 A4E           C=C-1  S              D(S) = 0
2511 084D3 94E           ?C#0   S
2512 084D6 37            GOYES  CRTIRM         D(S) >= 1  (PORT)
2513                * Amt of memory needed in C(A)
2514 084D8 8E00   CRETMF GOSUBL =MEMCKL         Do mem check with LEEWAY
           00
2515 084DE 400           RTNC                  Carry set=> not enough mem
2516                * Amt of memory needed (offset) in B(A)
2517                * A(A) contains AVMEMS (End Source)
2518                * D1 @ AVMEMS (2F5xx)
2519                *
2520                * Length of Source = End Source - Begin Source
2521                *                  = AVMEMS - (MAINEN - 2)
2522                *
2523 084E1 130     =CRETM5 DO=A                  AVMEMS
2524 084E4 1D17          D1=(2) =MAINEN
2525 084E8 147           C=DAT1 A
2526 084EB CE            C=C-1  A
2527 084ED CE            C=C-1  A              MAINEN - 2   (Bgn Source)
2528 084EF EE            C=A-C  A              Length of Source
2529                *
2530                * Set End of Source
2531                * Move memory down
2532                *
2533 084F1 C0            A=A+B  A              End Destination
2534 084F3 131           D1=A                  End Destination
2535 084F6 8E00          GOSUBL =MOved3
           00
2536                *
2537                * Zero flags & Write Time & Date to File header
2538                *   On Return from MOVED3, DO @ File start
2539                *
2540 084FC 7DD0          GOSUB  WFTMDT         WRITE FILE TIME
2541 08500 77B0          GOSUB  WFLENG         WRITE FILE LENGTH
2542                * Save values of PRGMEN & CURREN
2543 08504 1B76          DO=(5) =PRGMEN
           5F2
2544 0850B 146           C=DATO A
2545 0850E 06            RSTK=C                Save PRGMEN on stack
2546 08510 164           DO=DO+ (CURREN)-(PRGMEN)
2547 08513 142           A=DATO A              Read CURREN
2548                *
2549                * RFADJ WILL UPDATE PCADDR ONLY IF POINTING INTO I/OBUFFER
2550                *
2551 08516 121           AR1EX                 Bgn source in A(A); CURREN in R1
2552 08519 8E00          GOSUBL =RFADJ+        ADJUST FOR-NEXT/GOSUB STACKS
           00
2553                *
2554                * Restore PRGMEN & CURREN
```

```
2555                *
2556 0851F 119              C=R1                    Restore CURREN
2557 08522 1BC6             D0=(5) =CURREN
           5F2
2558 08529 144              DATO=C A                CURREN
2559 0852C 184              D0=D0- (CURREN)-(PRGMEN)
2560 0852F 07               C=RSTK                  Restore PRGMEN
2561 08531 144              DATO=C A                PRGMEN
2562 08534 8F00   rc01      GOSBVL =RC01            Restore R1 from WFTMDT
           000
2563 0853B 03               RTNCC                   RC01 returns w/ carry set
2564                *
2565                * Error Exits:
2566                *   Load Error number and Return with Carry Set
2567                *     Insufficient Memory to Create file  (RMEM)
2568                *     Port Device not found to create file
2569                *
2570 0853D 8C00   CRTEME GOLONG =RMEM10           "Insufficient Memory"
           00
2571                *
2572                * Carry is set
2573                *
2574 08543 8C00   PRTnFD GOLONG =FLDE90           LC(4) eDVCNF & rtn
           00
2575                *
2576                * PORT device type
2577                *   If no specific Port Number specified (D(B)=FF)
2578                *      Create File on first available port
2579                *
2580 08549 B67    CRTIRM D=D+1  B
2581 0854C 5C1              GONC   CRTI20           PORT SPECIFIED?
2582 0854F 8E00             GOSUBL =romchk
           00
2583 08555 4DE    CRTI10 GOC    PRTnFD            NO PORTS?
2584 08558 7610             GOSUB  CRTPRT           ATTEMPT TO CREATE FILE
2585 0855C 500              RTNNC                   DONE SUCCESSFULLY?
2586                *
2587                * UNABLE TO DO CREATE -
2588                *   EITHER TOO LITTLE MEMORY OR ILLEGAL MEDIA
2589                *
2590 0855F 8E00             GOSUBL =romfnd          SEARCH FOR NEXT PORT
           00
2591 08565 6FEF             GOTO   CRTI10
2592                *
2593                * PORT number specified
2594                *   Attempt to find specific port
2595                *
2596 08569 8E00   CRTI20 GOSUBL =ROMF-1
           00
2597 0856F 43D              GOC    PRTnFD           PORT NOT FOUND
2598                *
2599                * Try to Create File on PORT if IRAM port
2600                *
2601 08572 8E00   CRTPRT GOSUBL =RAMROM
           00
```

```
2602 08578 575          GONC   CRTI60          NOT RAM?
2603               *
2604               * D1 PAST FILE CHAIN POINTER
2605               *
2606 0857B 137          CD1EX
2607 0857E 8E00         GOSUBL =EOFLC+         CALCULATE WHERE CHAIN ENDS
           00
2608               * C(A) Points to end of file chain
2609 08584 E6           C=C+1  A               Pt 1 byte past end of file chain
2610 08586 E6           C=C+1  A               (past the 0 byte) - can't carry (NM
2611               * Now add proposed size of file
2612 08588 110          A=R0                   SIZE OF FILE
2613 0858B C2           C=A+C  A               Ptr past where 0 byte would be
2614 0858D 4FA          GOC    CRTEME          Guard against 'wrap-around'
2615 08590 06           RSTK=C                 Save ptr
2616 08592 8E00         GOSUBL =LSTADR         CALC. PAST LAST ADDR ON PORT
           00
2617               *
2618               * AVMEME FOR PORT IN C(A)
2619               *
2620 08598 DA           A=C    A
2621 0859A 07           C=RSTK
2622 0859C 8B2          ?C>A   A
2623 0859F E9           GOYES  CRTEME
2624 085A1 134          D0=C
2625 085A4 181          D0=D0- 2
2626 085A7 D2           C=0    A
2627 085A9 14C          DAT0=C B               WRITE OUT ZERO BYTE
2628 085AC 110          A=R0
2629 085AF 136          CD0EX
2630 085B2 E2           C=C-A  A
2631 085B4 134          D0=C
2632 085B7 7220         GOSUB  WFTMDT
2633               *
2634               * Compute File Chain Length
2635               *   File Chain Length = Total File size - Offset to File length i
2636               * Position to File length field   (D0 @ TIME field from WFTMDT)
2637               * Write File Length to Header
2638               *
2639 085BB D2    WFLENG C=0    A
2640 085BD 3102         LC(2)  =oFLENh         Offset to File length
2641 085C1 110          A=R0                   Total file size
2642 085C4 D8           B=A    A               Save Total size for ptr adjust
2643 085C6 EA           A=A-C  A               File Chain Length
2644 085C8 169          D0=D0+ (oFLENh)-(oTIMEh) Position to File Length field
2645 085CB 140          DAT0=A A               Write out new length
2646 085CE 03           RTNCC                  Successful CREATE
2647               *
2648               * Illegal Access Error
2649               *
2650 085D0 8C00  CRTI60 GOLONG =PRGFE          ILLEGAL (FILE) ACCESS
           00
```

```
2651                    STITLE Write TIME & DATE to file header
2652                ********************************************************************
2653                ********************************************************************
2654                **
2655                ** Name:(S) WFTMDT  -  Write Flags, Time, Date to File Header
2656                **
2657                ** Category:   FILUTL
2658                **
2659                ** Purpose:
2660                **      Zero Flags, Write Creation Time & Date to file header
2661                **
2662                ** Entry:
2663                **      D0 @ File start
2664                **
2665                **      WFTMD-:  Set flag to prevent Nib 2 of Flags to be zeroed
2666                **      WFTMDT:  Clear flag: Nib 2 of flags is zeroed
2667                **
2668                **      Nib 2 of flags = COPY code nibble
2669                **
2670                ** Exit:
2671                **      D0 @ Time field of file header
2672                **      P=0
2673                **      R1 @ File start
2674                **
2675                **      In RAM:
2676                **         Flag: 00
2677                **         Time: mmhh
2678                **         Date: ddmmyy
2679                **
2680                ** Calls:      STO1, YMDHMS, RCO1
2681                **
2682                ** Uses.......
2683                **  Exclusive: A(A),C,P,D0,R1
2684                **  Inclusive: A,B,C,D,P,D0,D1,R0,R1,SCRTCH (32 nibs),S0-S7
2685                **
2686                **      R1 = File start
2687                **      YMDHMS uses A-D,R0-R1,D0,D1,S0-S7
2688                **      STO1   uses A,D0,SCRTCH (32 nibs)
2689                **      RCO1   uses R0,R1,D0,A
2690                **
2691                ** Stk lvls:   3
2692                **
2693                ** Detail:
2694                **      STO1 called to save R0-R1 in SCRTCH
2695                **        YMDHMS uses these registers
2696                **        RCO1 restores R0-R1
2697                **
2698                ** NOTE:
2699                **      This routine could be shorter if another scratch
2700                **      register or the stack was used to save the position
2701                **      within the file header @ Time
2702                **
2703                **      Since this is a utility I'm trying to minimize the
2704                **      usage of R registers and subroutine levels
2705                **
```

```
2706                ** 2:   The positioning from the File start to the TIME field
2707                **      is through LENGTHs not OFFSETs.
2708                **
2709                ** History:
2710                **
2711                **    Date     Programmer              Modification
2712                **  --------   ----------   ----------------------------------
2713                **  07/04/82   JP           Modified documentation
2714                **
2715                **
2716                ************************************************************************
2717                ************************************************************************
2718 085D6 850    =WFTMD- ST=1   0              Set COPY nib flag
2719 085D9 6600         GOTO   WFTMD1
2720                *
2721                * Save File start
2722                * Adjust DO to Flag field in header
2723                * If COPY nib flag set
2724                *    Zero low nibble of flag field
2725                * else
2726                *    Zero both nibbles of flag field
2727                *
2728 085DD 840    =WFTMDT ST=0   0              Clear COPY nib flag
2729 085E0 132     WFTMD1 ADOEX                 DO @ File start
2730 085E3 101         R1=A                     Save File start
2731 085E6 D2          C=0    A
2732 085E8 3141        LC(2)  =oFLAGh           Offset to Flag field in header
2733 085EC CA          A=A+C  A
2734 085EE 130         DO=A                     Positioned @ Flag field
2735 085F1 D0          A=0    A
2736 085F3 870         ?ST=1  0                Zero low nibble of flag field ?
2737 085F6 40          GOYES  WFTMD4
2738 085F8 21          P=     1                Zero both flag nibbles
2739 085FA 1501   WFTMD4 DAT0=A WP              Zero Flags field
2740               *
2741               * P=0 gets done in YMDHMS subroutine
2742               *     ST01 does not care about P setting
2743               *
2744               * Save R0-R1
2745               * Get Date and Time
2746               * Shift off seconds in Time
2747               *
2748 085FE 8F00   WFTMD6 GOSBVL =ST01            Store R0 - R1
           000
2749 08605 8F00        GOSBVL =YMDHMS           Get Date and Time
           000
2750 0860C BF6         CSR    W                 Shift off Seconds
2751 0860F BF6         CSR    W
2752 08612 7E1F        GOSUB  rc01              Restore R0 - R1
2753               *                            (calling rc01 saves 1 nib)
2754               * Position to Time field
2755               * Write out Date, Time
2756               *
2757 08616 111         A=R1                     File start
2758 08619 130         DO=A
```

```
2759 0861C 16F          D0=D0+ lFNAMh          Skip Filename
2760 0861F 165          D0=D0+ (lFTYPh)+(lFLAGh)  Skip File Type, Flags
2761 08622 20           P=     0               Reset from YMDHMS
2762 08624 15C9         DAT0=C 10              Write Date,Time
2763 08628 03           RTNCC
```

```
2764                    STITLE EXCADR - Execution Address Compute
2765                ******************************************************************
2766                ******************************************************************
2767                **
2768                ** Name:    EXCADR  -  Compute Exec Addr of Token
2769                ** Name:(S) EXCAD+  -  Compute Exec Addr of Token
2770                **
2771                ** Category:   ADDCAL
2772                **
2773                ** Purpose:  Return Execution Address of Command Token,
2774                **           preserving D0,D1
2775                **
2776                ** Entry:    EXCADR: A(B) = Command token
2777                **                   Assumes MAIN Table in Mainframe
2778                **           EXCAD+: A(B) = Command token
2779                **                   C(A) = Main Table + 3 of XROM
2780                **                          Position @ Execution Address field
2781                **
2782                ** Exit:     C(A) = Execution Address for token
2783                **
2784                ** Calls:    None
2785                **
2786                ** Stk lvls: 0
2787                **
2788                ** Uses:     A(A),C(A)
2789                **
2790                ** Detail:   Preserves D0
2791                **           Address = Token * 9 + Main Table Adjustment
2792                **
2793                ** History:
2794                **
2795                **    Date     Programmer             Modification
2796                **  --------   ----------   ---------------------------------
2797                **  07/06/82   JP           Modified documentation
2798                **
2799                ******************************************************************
2800                ******************************************************************
2801                *
2802
2803 0862A 3400 =EXCADR LC(5)  (=MAINT)+3     Main Table Adjustment Addr
          000
2804                *                         Positions to Exec Addr field
2805 08631 C2   =EXCAD+ C=C+A  A              Add Table offset to Token
2806 08633 C4           A=A+A  A              Calc offset into Main Table
2807 08635 C4           A=A+A  A              Times 4
2808 08637 C4           A=A+A  A              9 nibs per table entry
2809 08639 CA           A=A+C  A              Add table start to offset
2810 0863B 132          ADOEX                 D0 <-- Addr of Exec addr, A <- Old
2811 0863E 146          C=DAT0 A              Read relative Exec Address
2812 08641 132          ADOEX                 Restore D0, A <- @ Exec Addr
2813 08644 C2           C=C+A  A              Absolute Exec Address
2814 08646 01           RTN
```

```
 ACTTMR  Ext                  -   303
 ALINFO  Ext                  -  1519
 ARGERR  Ext                  -   866
 ARGSTA  Ext                  -   245   858
 BASCHA  Ext                  -  2066
 BSERR   Ext                  -  1584
=BYE     Abs   32738 #07FE2   -   525
 CHAIN-  Ext                  -  2069
=COPY    Abs   33227 #081CB   -  1518  1514
 COPYDC  Ext                  -  1516
 COPYP   Ext                  -  1517
=COPYu   Abs   33385 #08269   -  1983  1628
 CPY010  Abs   33238 #081D6   -  1527  1614
 CPY020  Abs   33270 #081F6   -  1550  1566
 CPY030  Abs   33279 #081FF   -  1561  1539
 CPY040  Abs   33296 #08210   -  1572  1542  1563
 CPY050  Abs   33311 #0821F   -  1577  1565
 CPY060  Abs   33326 #0822E   -  1592  1574
 CPY065  Abs   33344 #08240   -  1603  1551  1593
 CPY070  Abs   33347 #08243   -  1610  1579
 CPY080  Abs   33363 #08253   -  1626  1612
 CPY095  Abs   33396 #08274   -  2003  2112
 CPY098  Abs   33436 #0829C   -  2015  2007
 CPY100  Abs   33470 #082BE   -  2043  1989
 CPY105  Abs   33490 #082D2   -  2049  2044  2047
 CPY110  Abs   33537 #08301   -  2075  2067
 CPY120  Abs   33551 #0830F   -  2085
 CPY130  Abs   33580 #0832C   -  2099  2088
 CPY135  Abs   33604 #08344   -  2106  2103
 CPY140  Abs   33624 #08358   -  2117  2108  2111
 CPY160  Abs   33641 #08369   -  2134  2118
 CPY170  Abs   33701 #083A5   -  2154  2144  2147
 CPY177  Abs   33719 #083B7   -  2166
 CPY180  Abs   33759 #083DF   -  2186  2245
 CPY220  Abs   33833 #08429   -  2230  2228
 CPY250  Abs   33459 #082B3   -  2034  2032
 CPY2CD  Abs   33440 #082A0   -  2027  2078
 CPYCRD  Abs   33809 #08411   -  2216  2048
 CPYER0  Abs   33424 #08290   -  2012  2009  2229
 CPYER1  Abs   33432 #08298   -  2014  2005  2050  2063
 CPYER2  Abs   33637 #08365   -  2124  2105  2149  2159
 CPYER3  Abs   33829 #08425   -  2229  2232
=CPYERR  Abs   33799 #08407   -  2206  2014  2124
 CPYERX  Abs   33320 #08228   -  1584  1520  1602  1629
 CPYEXT  Abs   33793 #08401   -  2202  2010  2015  2035  2190  2207
 CRDFIL  Ext                  -  2233
=CREATF  Abs   33985 #084C1   -  2505
=CRETF+  Abs   33988 #084C4   -  2506  2158
=CRETM5  Abs   34017 #084E1   -  2523
 CRETMF  Abs   34008 #084D8   -  2514  2509
 CRTEME  Abs   34109 #0853D   -  2570  2614  2623
 CRTI10  Abs   34133 #08555   -  2583  2591
 CRTI20  Abs   34153 #08569   -  2596  2581
 CRTI60  Abs   34256 #085D0   -  2650  2011  2602
 CRTIRM  Abs   34121 #08549   -  2580  2512
```

```
 CRTPRT  Abs   34162 #08572 -  2601  2584
 CURREN  Abs  193900 #2F56C -    12  2546  2557  2559
 D1=sR0  Abs   32637 #07F7D -   399   236   283  2100  2136
 DEFFIL  Ext                -  1550
 DSLEEP  Ext                -   529
 EOFLC+  Ext                -  2607
 ERRADR  Abs  194184 #2F688 -    12   219   484
 ERRSUB  Abs  194179 #2F683 -    12   216   219
=EXCAD+  Abs   34353 #08631 -  2805  1167
=EXCADR  Abs   34346 #0862A -  2803
=EXPER2  Abs   32954 #080BA -   866   391   860  1053
 EXPEX-  Ext                -   244
 EXPEXC  Ext                -   857
 EXPSKP  Ext                -   376
 FILCRD  Ext                -  2034
 FILFIL  Ext                -  1987
 FINDF   Ext                -  2249
=FINDFS  Abs   33864 #08448 -  2248  2049  2117
 FLDE90  Ext                -  2574
 FLDEV+  Ext                -  2102
=FLIP    Abs   32230 #07DE6 -    61
 FLIPDC  Ext                -    58
 FLIPP   Ext                -    59
 FSPECx  Ext                -  1601
 GETPR1  Ext                -  2062
 GOTO+   Ext                -   725
=GTMR#   Abs   32970 #080CA -  1043   235   505
 GTMR10  Abs   32989 #080DD -  1050  1048
=GTMRA+  Abs   33015 #080F7 -  1097   512
 GTMRA1  Abs   33037 #0810D -  1104  1096
=GTMRAD  Abs   33023 #080FF -  1100   286
=LC      Abs   32230 #07DE6 -    60
 LEXBF+  Ext                -  2193
 LNSKP-  Ext                -   645
 LSTADR  Ext                -  2616
 LXFND   Ext                -  1247
 MAINEN  Abs  193905 #2F571 -    12  2524
 MAINT   Ext                -  1352  2803
 MEMCKL  Ext                -  2514
 MFDEVC  Ext                -  2045
 MFDVC+  Ext                -  2076  2134  2218
 MFERR   Ext                -   869
 MOVEU0  Ext                -  2170
 MOved3  Ext                -  2535
=MTADDR  Abs   33173 #08195 -  1352
=MTADR+  Abs   33185 #081A1 -  1355
 Mferr   Abs   32964 #080C4 -   869  1052
 NXTSTM  Ext                -  1630
 Nxtstm  Abs   33375 #0825F -  1630    83   223
=OFF     Abs   32656 #07F90 -   476
 OFF10   Abs   32678 #07FA6 -   485   513
 OFF20   Abs   32691 #07FB3 -   502   483
 OFF30   Abs   32738 #07FE2 -   526   504
 OFF40   Abs   32749 #07FED -   529   527
 OFFDC   Ext                -   474
```

```
 OFFEXT  Abs   32755 #07FF3 -   530   489
 OFFP    Ext                 -   475
=OFFTMR  Abs   32702 #07FBE -   506
=ON      Abs   32308 #07E34 -   208
 ON010   Abs   32348 #07E5C -   227   211
 ON015   Abs   32360 #07E68 -   235   229
 ON030   Abs   32396 #07E8C -   254   263
 ON035   Abs   32401 #07E91 -   260   252
 ON040   Abs   32432 #07EB0 -   277   255   272
 ON300   Abs   32485 #07EE5 -   310   230
 ON400   Abs   32526 #07F0E -   325   322
 ON500   Abs   32529 #07F11 -   329   319   324   386
 ON700   Abs   32558 #07F2E -   353   338
 ON800   Abs   32582 #07F46 -   362   365
 ON850   Abs   32599 #07F57 -   374   360
 ON900   Abs   32618 #07F6A -   384   341   355   367
 ON999   Abs   32630 #07F76 -   395   331
 ONDC    Ext                 -   206
 ONER1   Abs   32626 #07F72 -   391
=ONERR   Abs   32759 #07FF7 -   636
 ONEXIT  Abs   32344 #07E58 -   223   304   530
 ONGTGB  Abs   32876 #0806C -   724   396
 ONP     Ext                 -   207
=ONTIMR  Abs   32776 #08008 -   663
 ONTM20  Abs   32834 #08042 -   696   675   686
 ONTM25  Abs   32847 #0804F -   704   697
 ONTM30  Abs   32870 #08066 -   722   712
 POllj   Ext                 -  2250
 PRGFE   Ext                 -  2650
 PRGMEN  Abs  193895 #2F567 -    12  2543  2546  2559
 PRTnFD  Abs   34115 #08543 -  2574  2583  2597
 PUTALM  Ext                 -   511
 PWROFF  Ext                 -   528
 PgmRun  Abs      13 #0000D -    11   526
 Poll    Abs   33873 #08451 -  2250  1050  2003
=RACTMI  Abs   32883 #08073 -   782   687
 RAMROM  Ext                 -  2601
 RC01    Ext                 -  2562
 RCTTM1  Ext                 -   794
 RDHDR1  Ext                 -  2140
 RDIN10  Abs   33902 #0846E -  2355  2350
 RDIN20  Abs   33932 #0848C -  2370  2367
 RDIN30  Abs   33961 #084A9 -  2386  2371
=RDINFD  Abs   33896 #08468 -  2353  2176
=RDINFO  Abs   33899 #0846B -  2354  2352
=RDINFS  Abs   33889 #08461 -  2351
=RENAME  Abs   33210 #081BA -  1513
 RENAMP  Ext                 -  1512
 RENMDC  Ext                 -  1511
 RFADJ+  Ext                 -  2552
 RLINFO  Ext                 -  2202
 RMEM10  Ext                 -  2570
=RNDEXP  Abs   32928 #080A0 -   857   312  1044
 RNM010  Ext                 -  1632
 ROMF-1  Ext                 -  2596
```

RTNCC	Ext			-	518				
Range	Ext			-	1276				
SAVSTK	Abs	193950	#2F59E	-	12	2355			
SEC2TK	Ext			-	260				
SFLAG*	Ext			-	93				
SFLAGT	Ext			-	82				
ST01	Ext			-	2748				
STMTRO	Abs	194673	#2F871	-	12	399			
STOPDC	Ext			-	517				
=SVINF+	Abs	33879	#08457	-	2348				
=SVINFO	Abs	33882	#0845A	-	2349	1610			
=TMIADR	Abs	33001	#080E9	-	1094	293	790		
TMRAD1	Abs	194199	#2F697	-	12	1100			
TMRIN1	Abs	194214	#2F6A6	-	12	1094			
TRFROM	Ext			-	698				
TRTO*	Ext			-	713				
TrfCk-	Ext			-	696	711			
UPDPC	Ext			-	706				
=USER	Abs	32248	#07DF8	-	65				
USER10	Abs	32252	#07DFC	-	66	62			
USER20	Abs	32286	#07E1E	-	87	71			
USER30	Abs	32288	#07E20	-	93	78			
USERDC	Ext			-	63				
USERP	Ext			-	64				
USREXT	Abs	32282	#07E1A	-	83	94			
WFLENG	Abs	34235	#085BB	-	2639	2541			
=WFTMD-	Abs	34262	#085D6	-	2718	2180			
WFTMD1	Abs	34272	#085E0	-	2729	2719			
WFTMD4	Abs	34298	#085FA	-	2739	2737			
WFTMD6	Abs	34302	#085FE	-	2748				
=WFTMDT	Abs	34269	#085DD	-	2728	2540	2632		
XMTA10	Abs	33096	#08148	-	1257	1296			
XMTA15	Abs	33150	#0817E	-	1293	1278			
XMTA20	Abs	33154	#08182	-	1295	1261			
XMTA25	Abs	33160	#08188	-	1304	1259			
XMTA30	Abs	33162	#0818A	-	1305	1248	1286		
=XMTADR	Abs	33075	#08133	-	1241	1160			
XWDERR	Abs	32960	#080C0	-	868	1161			
=XWORD	Abs	33046	#08116	-	1159				
YMDHMS	Ext			-	2749				
dCARD	Abs	7	#00007	-	11	1575			
dIRAM	Abs	1	#00001	-	11	2109			
dPORT	Abs	1	#00001	-	11	2230			
eFEXST	Ext			-	2123				
eFSPEC	Ext			-	2013				
eFTYPE	Ext			-	2148				
eXWORD	Ext			-	868				
fKEY	Abs	57868	#0E20C	-	11	2145			
fLEX	Abs	57864	#0E208	-	11	2188			
flLC	Abs	-15	#FFFF1	-	11	61			
flUSER	Abs	-9	#FFFF7	-	11	65			
fltdh	Ext			-	859				
lDEVC	Abs	5	#00005	-	11	2361	2368		
lFLAGh	Abs	2	#00002	-	11	2760			
lFNAM+	Abs	4	#00004	-	11	2361	2368	2379	2390

```
lFNAM8  Abs      16 #00010 -    11  2362  2369  2376  2387
lFNAMh  Abs      16 #00010 -    11  2759
lFTYPh  Abs       4 #00004 -    11  2760
lLXENT  Abs      11 #0000B -    11  1295
lLXID   Abs       2 #00002 -    11  1257  1284
lLXTKR  Abs       4 #00004 -    11  1257  1284
oFLAGh  Abs      20 #00014 -    11  2732
oFLENh  Abs      32 #00020 -    11  2156  2640  2644
oFTYPh  Abs      16 #00010 -    11  2185  2244
oTIMEh  Abs      22 #00016 -    11  2185  2644
pCOPYx  Abs       8 #00008 -    11  2004
pTIMR#  Abs      59 #0003B -    11  1051
rc01    Abs   34100 #08534 -  2562  2752
renam   Abs   33379 #08263 -  1632  1627
romchk  Ext         -  2582
romfnd  Ext         -  2590
sCARD   Abs       2 #00002 -    11  2043  2046  2077  2227
sDEST   Abs       3 #00003 -    11  1541  1564  1611  1613  2075  2217  2351
                              2353  2366
sEXTDV  Abs       0 #00000 -    11  1988
sEXTGS  Abs       5 #00005 -    11   395   476   487   664
sGOSUB  Abs       3 #00003 -    11   310   325   671   684
sKEYS   Abs       5 #00005 -    11  2085  2089  2143
sONERR  Abs       4 #00004 -    11   637
sONTMR  Abs       6 #00006 -    11   685
sPCRD   Abs       8 #00008 -    11  2029  2033
sREADI  Abs       4 #00004 -    11  2349  2354  2370
sRENAM  Abs       6 #00006 -    11  1513  1626
sRESTR  Abs      10 #0000A -    11   311   323   663
sXWORD  Abs       9 #00009 -    11   724
tCARD   Ext         -  1572
tERROR  Ext         -   209   481
tGOTO   Ext         -   317   673
tKYSck  Ext         -  1592
tLINE#  Ext         -   336
tLITRL  Ext         -   358
tON     Ext         -    67
tTO     Ext         -  1537
```

Input Parameters

  Source file name is JP&EXC::MS

  Listing file name is JP/EXC:TI:ML::-1

  Object file name is JP%EXC:TI:MS::-1

                                111111
                       0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS    GGG    &     EEEEE  X   X   CCC
 2            *    S   S  G   G  & &    E      X   X  C   C
 3            *    S      G      & &    E       X X   C
 4            *     SSS   G GGG   &     EEEE     X    C
 5            *        S  G   G  & & &  E       X X   C
 6            *    S   S  G   G  &  &   E      X   X  C   C
 7            *     SSS    GGG    && &  EEEEE  X   X   CCC
 8            *
 9                      TITLE  Miscellaneous Execution Routines<831216.1613>
10 08648               ABS    #8648
11                     RDSYMB TI%EQU::MS
12                     RDSYMB SB%RAM::MS
13           *
14           ******************************************************************
15           ******************************************************************
16           **
17           ** Name:    FOR     -  Executes FOR statement
18           **
19           ** Category: STEXEC
20           **
21           ** Purpose:  EXECUTES 'FOR' STATEMENT
22           **
23           ** Entry:    DO points past tFOR
24           **           P=0
25           **
26           ** Exit:     2 CASES: 1) IF the loop is to be executed, DO
27           **                       points to the end of the stmt.
28           **                       The following data has been pushed
29           **                       on the FOR/NEXT stack:
30           **
31           **           FORSTK =>    RETURN ADDRESS (DO)           5 NIBS
32           **                       STEP VALUE                   16 NIBS
33           **                       LIMIT                        16 NIBS
34           **                       ASCII LETTER OR ALPHA-DIGIT TK 2 NIBS
35           **                       00 OR ASCII LETTER            2 NIBS
36           **
37           **
38           **                    2) Initial conditions dictate that the
39           **                       loop isn't be executed at all. Program
40           **                       memory (or in the case of CALC BASIC,
41           **                       the line until tEOL) is scanned,
42           **                       searching for a tNEXT with a matching
43           **                       index variable.
44           **                       If no match is found, eFWONX is given.
45           **
46           ** Calls:    ASNMNT, NXTCK9, POPSH, LINSKP, TRFLCK, TRCFRM,
47           **           TRTO-, INDXSN, TKSCN7, POPSTK, GETST, RSTST, uTST12,
48           **           EOLSCN, TRCVAR, INVRES, DO=PCR, XQBLK?, MEMCKL
49           **
50           ** Uses:     A-D, P, R0, R2, D0, D1, S2, S8
51           **            R1, S9, S-R1-2 - (only if TRACE is on)
52           **            + R1,R3, all of function scratch, S0-S11 - EXPEXC
53           **
54           ** Detail:   FOR stack is searched for an entry with same index.
55           **           If one is found, that entry is deleted from the
```

```
                **          stack, and the new FOR declaration is pushed on the
                **          revised top of the FOR-NEXT stack.  [AS HANDLED BY
                **          CAPRICORN & POINTED OUT IN BYTE MAGAZINE (1-81)].
                **
                **          When a loop descripter is initially pushed on the
                **          FOR-NEXT stack, 52 nibbles are allocated.  This is
                **          because after the 4 nibble index is pushed on the
                **          stack, the assignment value is pushed.  Everything
                **          else follows, except the return address; it isn't
                **          pushed until it is certain the loop will be
                **          executed and the assignment value deleted from the
                **          stack.  At that time, 16 nibbles are released from
                **          the stack and therefore returned to available
                **          memory.
                **
                ** Stack lvls: 6
                **
                ** History:
                **
                **    Date     Programmer   Modifications
                ** --------   ----------   -------------
                ** 10/03/82   S.C.         Replace GOSUB PCCHK by ?S13=1
                **                          assuming S13 is valid
                ** 06/30/82   S.W.         Added documentation.
                ** 07/07/82   S.W.         In FRNULL code, eliminated
                **                          CURRL update.
                ** 10/15/82   S.W.         Replaced calls to INVNaN &
                **                          uRES12 with a single call to
                **                          INVRES (per PM)
                ** 07/06/83   S.W.         Check for complex index var.
                **
                ********************************************************************
                ********************************************************************
                *
08648 6C94  FORXQe GOTO   rdatty          Data Type error
                *
0864C 6706  FORERR GOTO   mferr           Insufficient mem
                *
08650 0000         REL(5) =FORDC
      0
08655 0000         REL(5) =FORP
      0
0865A 7C13 =FOR    GOSUB  INDXSN
0865E D9           C=B    A
08660 06           RSTK=C                 Save variable name
08662 582          GONC   FOR10          NO MATCH?
                * MATCH FOUND - TRIM STACK
08665 136          CDOEX
08668 06           RSTK=C                 SAVE PC
0866A 173          D1=D1+ 4              BLOCK END
0866D 133          AD1EX
08670 D2           C=0    A
08672 3192         LC(2)  41
08676 EE           C=A-C  A              BLOCK START
08678 21           P=     1
```

```
109 0867A 8E5D          GOSUBL POPSTK
          80
110 08680 07            C=RSTK
111 08682 134           DO=C                    RESTORE PC
112 08685 133           AD1EX
113 08688 100           RO=A
114               *
115 0868B D2    FOR10   C=0     A
116 0868D 3143          LCHEX   34              Var name, init. val, limit, step
117 08691 8E00          GOSUBL  =MEMCKL         Check with leeway
          00
118 08697 44B           GOC     FORERR
119 0869A 07            C=RSTK
120 0869C D5            B=C     A               Restore variable name
121 0869E 118           C=RO                    RESTORE 'TRUE' FORSTK
122 086A1 135           D1=C                    ACTUAL TOP OF FOR-NEXT STACK
123 086A4 1C3           D1=D1- 4
124 086A7 7982          GOSUB   NXTCK9          UPDATE FORSTK
125 086AB D4            A=B     A
126 086AD 1593          DAT1=A 4                PUSH VBL ON FOR-NXT STK
127 086B1 8E00          GOSUBL  =GETST          SAVE STATUSES (S8,S2-sENDx)
          00
128 086B7 8E00          GOSUBL  =ASNMNT         DO ASSIGNMENT OF EXPR TO VBL
          00
129 086BD 30E           LCHEX   E
130 086C0 902           ?A=C    P               Complex?
131 086C3 58            GOYES   FORXQe
132               * Update top of FOR-NEXT stack pointer
133 086C5 7B62          GOSUB   NXTCK9          Update FORSTK
134               *
135               * Returns w/ DO pointing at tTO
136               *
137 086C9 7603          GOSUB   POPSH           Throw limit on math stack
138               *
139               * Returns w/ D1 pointing to limit on FOR-NEXT stack
140               *  & DO pointing at tSTEP or statement end token
141               *
142 086CD 14A           A=DAT0 B
143 086D0 3100          LC(2)   =tSTEP
144 086D4 966           ?A#C    B               NOT STEP TOKEN?
145 086D7 A1            GOYES   FRXQ30
146 086D9 76F2          GOSUB   POPSH           PUSH STEP ON FOR-NEXT STACK
147
148
149
150
151               *.....................................................*
152               *  Check for an unordered STEP size (i.e. NaN )
153               *  and record this info. in st. bit UNSTEP. -- J.T.          *
154                                                                          .
155               UNSTEP  EQU     2                                          .
156                                                                          .
157 086DD 05            SETDEC                                             .
158 086DF AF2           C=0     W               A=STEP, C=0                .
159 086E2 28            P=      8               P="unord"                  .
```

```
    160 086E4 7882        GOSUB  utst12         STEP unord with 0 ? (CARRY=> TRUE)
    161 086E8 5C1         GONC   FRXQ35         (STEP is ordered )              .
    162                                                                         .
    163 086EB 852         ST=1   UNSTEP         STEP unord. with 0              .
    164 086EE 491         GOC    FRXQ40         (B.E.T.)                        .
    165                                                                         .
    166             *.....................................................*
    167             *
    168             *  DEFAULT STEP VALUE IS +1
    169 086F1 AF0   FRXQ30 A=0    W              DO POINTS AT EOL OR @
    170 086F4 B44         A=A+1  S
    171 086F7 BF4         ASR    W
    172 086FA 1CF         D1=D1- 16
    173 086FD 1517        DAT1=A W              PUSH STEP VALUE OF 1
    174 08701 7F22        GOSUB  NXTCK9         UPDATE FORSTK
    175             * READ INITIAL VALUE INTO D(W); THEN COLLAPSE THAT ENTRY IN STACK
    176 08705 842   FRXQ35 ST=0   UNSTEP         STEP ordered
    177 08708 17F   FRXQ40 D1=D1+ 16
    178 0870B 17F         D1=D1+ 16
    179             *
    180 0870E 133         AD1EX
    181 08711 131         D1=A                  START OF BLOCK TO DELETE
    182             *
    183 08714 1577        C=DAT1 W              POP ASSIGNMENT VAL. INTO C
    184 08718 AF7         D=C    W
    185 0871B 17F         D1=D1+ 16             PT D1 @ VAR NAME - END OF BLOCK
    186 0871E 137         CD1EX
    187 08721 DE          ACEX   A
    188             *
    189             * NOW CLOSE MEMORY TO DELETE ASSIGNMENT VALUE FROM FOR-NEXT STACK
    190             *
    191 08723 21          P=     1              ADJUST FORSTK & AVMEME POINTERS
    192 08725 8EA2        GOSUBL POPSTK         CLOSE 'GAP'
              80
    193             *
    194             * D1 PTS TO STEP VALUE
    195             *
    196 0872B AFB         C=D    W              RESTORE ASSIGNMENT VALUE
    197             *.....................................................*
    198             *   Do the initial testing for the loop. -- J.T.      .
    199             *                                                     .
    200             *     1) If UNSTEP=1 then goto UNORD                  .
    201             *     2) Test (INDEX=LIMIT)*SGN(STEP) =< 0            .
    202             *                                                     .
    203             *          if "=<" then execute block.                .
    204             *          if ">" then exit loop.                     .
    205             *          if "unord" then goto UNORD.                .
    206                                                                   .
    207                                                                   .
    208 0872E 872         ?ST=1  UNSTEP                                  .
    209 08731 41          GOYES  UNORD                                   .
    210 08733 AF5         B=C    W
    211 08736 7212        GOSUB  XQBLK?         execute block? (i.e. "=<")    .
    212             * D1 POINTS TO LIMIT VALUE ON STACK
    213 0873A 473         GOC    FRXQ50         GO WITH CARRY SET             .
```

```
214 0873D 888          ?P#    8              ordered compare (i.e. ">")    .
215 08740 E2           GOYES  FRXQ45                                       .
216 08742 1CF          D1=D1- 16
217                                                                        .
218            *   Unordered parameter in FOR statement   *                .
219                                                                        .
220 08745      UNORD                                                       .
221 08745 05           SETDEC                                              .
222 08747 20           P=     0                                            .
223 08749 3100         LC(2)  =eUNORC        unord. compare msg.           .
224 0874D 8E00         GOSUBL =INVRES        signal IV oper.
          00
225 08753 AFA          A=C    W              A=NaN                         .
226 08756 137          CD1EX
227 08759 06           RSTK=C                SAVE D1
228            * SET UP S-R1-2 IN CASE OF TRACE
229            * and call STORE
230 0875B 7691         GOSUB  TRCVAR
231 0875F 07           C=RSTK
232 08761 135          D1=C                  POINTING TO STEP
233 08764 17F          D1=D1+ 16             POINT TO LIMIT
234 08767 D2           C=0    A
235 08769 96A          ?C=0   B              (B.E.T.)                      .
236 0876C 60           GOYES  FRXQ50         goes with carry set (since C=NaN)
237                                                                        .
238 0876E 7CB1 FRXQ45  GOSUB  NXTCK7         clears carry                  .
239            *.....................................................*
240            * DOESN'T AFFECT CARRY
241            *
242            * Following routine CAN'T affect carry
243 08772 8E00 FRXQ50  GOSUBL =DO=PCA
          00
244            *
245 08778 571          GONC   FRNULL         CARRY CLR=>DON'T DO LOOP
246            * GET ADDRESS OF NEXT LINE
247 0877B 7682         GOSUB  LINSKP
248 0877F 1CF          D1=D1- 16
249 08782 1C4          D1=D1- 5              POINT TO WHERE RTN ADDR TO GO
250 08785 141          DAT1=A A              PUSH 'RTN' ADDR ON FOR-NEXT STACK
251 08788 78A1         GOSUB  NXTCK9         UPDATE FORSTK
252 0878C 6951         GOTO   NXTST5
253            *
254            *
255            * FOR loop not to be executed at all; entry on FOR-NEXT stack is
256            *  already collapsed. D1 points at top of FOR-NEXT stack
257            *
258 08790 8E00  FRNULL GOSUBL =RSTST         REST.STATUSES (S8,S2-sENDx)
          00
259 08796 163          D0=D0+ 4              PT TO INDEX VBL
260 08799 146          C=DAT0 A
261 0879C 108          R0=C                  SAVE INDEX VAR TO SCAN FOR
262 0879F 1F76         D1=(5) =PRGMEN        GET END OF PROGRAM TO SEARCH
          5F2
263 087A6 147          C=DAT1 A
264 087A9 D7           D=C    A
```

```
265           * DETERMINE IF RUNNING FROM A PROGRAM
266 087AB 136         CDOEX
267 087AE 06          RSTK=C                SAVE D0
268 087B0 87D         ?ST=1  13             RUNNING FROM PROGRAM ?
269 087B3 E0          GOYES  FRNL04         RUNNING FROM PROGRAM?
270           * RUNNING FROM STATEMENT BUFFER - SET D(A) PROPERLY
271 087B5 7EE2        GOSUB  EOLSCN
272 087B9 161         D0=D0+ 2
273 087BC 136         CDOEX
274 087BF D7          D=C    A
275           *
276 087C1 07   FRNL04 C=RSTK
277 087C3 134         D0=C                  RESTORE D0
278 087C6 23          P=     3
279 087C8 878         ?ST=1  8
280 087CB 40          GOYES  FRNL05
281 087CD 21          P=     1
282 087CF 80FF FRNL05 CPEX   15
283           *
284 087D3 183  FRNL07 D0=D0- 4              PT TO LENGTH BYTE
285 087D6 7B22        GOSUB  LINSKP         & SCAN TO EOL (OR @)
286 087DA 20          P=     0
287 087DC 3100        LC(2)  =tNEXT         'NEXT TOKEN'
288           *
289           * SETS S9 TO SAVE LINE#, INITIALIZES A(A)=0 & F-R0-0
290           *
291 087E0 75B2        GOSUB  TKSCN7         SCAN FOR NEXT 'NEXT' STMT
292 087E4 4A0         GOC    NXTSN7         CARRY=>MATCH FOUND
293 087E7 3100        LC(2)  =eFwoNX        FOR WITHOUT NEXT
294 087EB 6864        GOTO   mferr
295           *
296 087EF 80DF NXTSN7 P=C    15             RESTORE P
297 087F3 110         A=R0                  RESTORE INDEX VBL OF FOR STMT
298 087F6 161         D0=D0+ 2              PT TO INDEX VBL OF NEXT STMT
299 087F9 1561        C=DAT0 WP
300 087FD 916         ?A#C   WP             INDEX VBLS DON'T MATCH?
301 08800 3D          GOYES  FRNL07
302           * FOUND NEXT W/MATCHING INDEX VBL
303           * CHECK IF NEED TO TRACE FLOW
304 08802 183         D0=D0- 4
305 08805 7364        GOSUB  trflck
306 08809 401         GOC    FRNL20         IF NO NEED TO TRACE, FRNL20
307 0880C 7844        GOSUB  trcfrm         TRACE THE FROM LINE(THE FOR LINE)
308 08810 8E00        GOSUBL =TRTO-         TRACE THE TO LINE(THE NEXT LINE)
          00
309 08816 7A44        GOSUB  trcfr1
310           *
311 0881A 6D32 FRNL20 GOTO   NXTST2
312           *
313           *
314           ********************************************************************
315           ********************************************************************
316           **
317           ** Name:    NEXT    -  Executes NEXT statement
318           **
```

```
                ** Category:  STEXEC
                **
                ** Purpose:  Executes NEXT statement
                **
                ** Entry:    D0 points to index variable (past tNEXT)
                **           P=0
                **
                ** Exit:     2 CASES: 1) LOOP IS REPEATED.  D0 contains return
                **                        address found on FOR/NEXT stack.
                **
                **                  2) LOOP IS TERMINATED.  Data on FOR-NEXT
                **                     stack pertaining to that index variable
                **                     is popped off the stack & control
                **                     goes to NXTSTM.
                **
                ** Calls:    NXTCK9, TRFROM, TRTO+, TRFLCK, STMBFD, D1FSTK,
                **           uAD12, INDXSN, EXPEXC, POP1R, XQBLK?, TRCVAR
                **
                ** Uses:     A-D, R0, S2, S8, D1, D0, S-R0-0
                **           If TRACE is on, additionally use: S9,R1,R2,S-R1-2
                **            + R0-R3, all of function scratch, S0-S11 - EXPEXC
                **
                ** Detail:   HOOKS IN FOR TRACE
                **
                ** Stack lvls: 6
                **
                ** History:
                **
                **    Date     Programmer   Modifications
                **  --------   ----------   -------------
                **  06/30/82   S.W.         Added documentation
                **  05/18/83   S.W.         NEXT in running program must
                **                           check for collapsed statement
                **                           buffer - See SR#698-1
                **
                ***************************************************************
                ***************************************************************
                *
                * NEXT from keyboard; ensure that address on stack points within
                * the statement buffer.  If FOR issued in program and program
                * suspended, do not want NEXT to start executing inside program
0881E 06     NEXTKB RSTK=C                 Save on stack
08820 7897          GOSUB  stmbfd          Locate start of stmt buffer
08824 D8            B=A    A               Save buffer length
08826 133           AD1EX                  Stmt Buf Start
08829 07            C=RSTK                 'Rtn' address
0882B 8B6           ?C<A   A               Addr prior to stmt buffer start?
0882E C0            GOYES  NXTE32
08830 C0            A=A+B  A               Points past end of stmt buffer
08832 8BA           ?C>=A  A               Address after end of buffer?
08835 50            GOYES  NXTE32
08837 553           GONC   NEXT10          (B.E.T.)
                *
0883A 3100   NXTE32 LC(2)  =eNXmoF
0883E 6514          GOTO   nferr
```

```
374                *
375 08842 0000             REL(5) =NXTDC
          0
376 08847 0000             REL(5) =NXTP
          0
377 0884C 7A21     =NEXT   GOSUB  INDXSN
378 08850 59E              GONC   NXTE32          NO MATCH?
379                * D1 POINTS TO VARIABLE ON FOR-NEXT STACK
380 08853 1CF              D1=D1- 16
381 08856 1CF              D1=D1- 16
382 08859 1C4              D1=D1- 5
383 0885C 74D0             GOSUB  NXTCK9          UPDATE AVMEME & FORSTK FROM D1
384                * D1 pointing to 'Rtn' Address
385 08860 147              C=DAT1 A
386 08863 8AA              ?C=0   A
387 08866 4D               GOYES  NXTE32          Collapsed in STMT buffer?
388 08868 86D              ?ST=0  13              NEXT from keyboard?
389 0886B 3B               GOYES  NEXTKB
390                * Holding out on calling 'expexc' for speed sake
391 0886D 8E00      NEXT10 GOSUBL =EXPEXC
          00
392 08873 AF9              C=B    W               Save var type & addr
393 08876 8E00             GOSUBL =POP1R          ENSURE IT'S REAL
          00
394                * RETURNS IN DEC MODE
395                *
396                * VARIABLE VALUE IN A
397                * C(S)=TYPE; C(A)=VBL ADDR
398                *
399 0887C 1B17             D0=(5) =S-R0-0
          8F2
400 08883 1547             DAT0=C W               PREPARE FOR CALL TO 'STORE'
401                *
402 08887 1AE9             D0=(4) =FORSTK
          5F
403 0888D 146              C=DAT0 A
404 08890 134              D0=C
405 08893 164              D0=D0+ 5
406 08896 1567             C=DAT0 W               STEP IN C
407                * D1 is intact from EXPEXC (points to end of avail mem)
408                * This is necessary in case of warning
409 0889A 8E00             GOSUBL =uAD12          C= VBL + STEP
          00
410                * C IS THE VALUE TO STORE IN INDEX VARIABLE
411 088A0 AFA              A=C    W
412                * TRCVAR will:
413                * 1) SETHEX
414                * 2) Save variable address in case of TRACE
415                * 3) Push variable on math stack - always needed by TRACE;
416                *    needed by STORE if variable doesn't exist
417 088A3 7E40             GOSUB  TRCVAR          STORE PTR TO VAR NAME
418                * STORE DESTROYS C
419 088A7 113              A=R3                   Recall value stored
420 088AA AF8              B=A    W
421                *
```

```
422            * B     = value of INDEX VBL
423            *
424 088AD 8F00         GOSBVL =D1FSTK        Position D1 to top of stk
          000
425 088B4 174          D1=D1+ 5              POINT TO STEP VALUE
426 088B7 7190         GOSUB  XQBLK?
427 088BB 513          GONC   NXTDNE
428            * CONTINUE LOOP - D1 POINTS AT LIMIT ON STACK
429 088BE 1CF          D1=D1- 16
430 088C1 1C4          D1=D1- 5              D1 PTS AT 'RTN' ADDRESS
431 088C4 143          A=DAT1 A              RTN ADDRESS IN A
432            *
433 088C7 86F          ?ST=0  15             Not in TRACE mode?
434 088CA 91           GOYES  NEXT55         YES => No TRACE
435            *
436            * CHECK TRACE MODE
437            *
438 088CC 130          DO=A
439 088CF 7993         GOSUB  trflck         Traps out execute from KB
440 088D3 421          GOC    NXTST5         NO NEED TO TRACE
441            * TRACE - don't worry about speed
442 088D6 7E73         GOSUB  trcfrm
443 088DA 8E00         GOSUBL =TRTO+
          00
444 088E0 112   NEXT50 A=R2
445 088E3 130   NEXT55 DO=A
446 088E6 841   NXTST5 ST=0   sENDx
447 088E9 6B71         GOTO   NXTST3
448            *
449            *      LOOP DONE - COLLAPSE STACK ENTRY
450 088ED 7D30  NXTDNE GOSUB  NXTCK7         COLLAPSE STACK ENTRY
451 088F1 6651         GOTO   NXTSTM
452            *
453            *
454            * VT      DT
455            * --      --
456            * A       2           INTEGER
457            * B       1           SHORT
458            * C      xxx          REAL (ARRAY)
459            * D       F           STRING
460            * 0-9     0           REAL (SCALAR)
461            *
462            *         E           COMPLEX
463            *
464            * PERTAINING TO VBL HEADER: IF DIG. 0>9 & DIG. 1<>0 => ARRAY
465            *
466            *
467            ********************************************************************
468            ********************************************************************
469            **
470            ** Name:    TRCVAR  -  trace variable
471            **
472            ** Category:   LOCAL
473            **
474            ** Purpose:
```

```
475                **      Prepares for call to STORE & possible TRACE
476                **      SETHEX
477                **      Writes out address of variable name in S-R1-2 and pushes
478                **      variable assignment value on stack, in preparation for
479                **      possible TRACE in STORE. Calls STORE
480                **
481                ** Entry:
482                **      PCADDR points at statement length of FOR or NEXT stmt.
483                **      A(W) contains value to be stored
484                **
485                ** Exit:
486                **      S-R1-2 as described above
487                **      Index variable value on math stack
488                **      HEX mode
489                **      P= 0
490                **      via STORE
491                **
492                ** Calls:      C=MSTK, DT1=C
493                **
494                ** Uses.......
495                **  Inclusive: C(A), D1, S-R1-2
496                **
497                ** Stk lvls:   2
498                **
499                ** History:
500                **
501                **    Date      Programmer              Modification
502                **  --------   ----------   ----------------------------------
503                **  06/30/82   S.W.         Added documentation
504                **  08/10/82   S.W.         Value now pushed on stack
505                **  10/11/82   S.W.         Put in call to D1FSTK
506                **  12/09/82   S.W.         exits via STORE
507                **
508                ********************************************************************
509                ********************************************************************
510 088F5 04     TRCVAR SETHEX
511                * Next 2 instructions may be eliminated to save code
512                * A time saver
513 088F7 86F            ?ST=0  15              Not in TRACE mode?
514 088FA F1             GOYES  TRCVR5
515 088FC 1F97           D1=(5) =PCADDR
           6F2
516 08903 147            C=DAT1 A
517 08906 135            D1=C
518 08909 173            D1=D1+ 4               STEP OVER STMT LENGTH & TOKEN
519 0890C 137            CD1EX
520 0890F 1FB8           D1=(5) =S-R1-2
           8F2
521 08916 145            DAT1=C A
522                *
523 08919 73D0   TRCVR5 GOSUB  C=MSTK           Position C(A) to top of math stack
524                * Can later cut some code (13 nibs) by putting some
525                * overhead in NXTCK9, so MTHSTK will be updated already
526 0891D 1C4            D1=D1- 5               Same as D1=(5) =MTHSTK
527 08920 74C0           GOSUB  DT1=C           UPDATE MTHSTK PTR & Pos D1 to it
```

```
528 08924 1517          DAT1=A W              PUSH ON STACK
529 08928 8C00          GOLONG =STORE
          00
530             *
531             *
532             ********************************************************************
533             ********************************************************************
534             **
535             ** Name:    NXTCK9  -  NEXT Check 9
536             **
537             ** Category:   LOCAL
538             **
539             ** Purpose:  Collapses FOR/NEXT stack entry; updates FORSTK
540             **
541             ** Entry:    2 ENTRY POINTS
542             **             1) NXTCK7 - ASSUMES D1 POINTS TO LIMIT VALUE ON
543             **                         STACK - COLLAPSES THAT ENTRY & UPDATES
544             **                         FORSTK.
545             **
546             **             2) NXTCK9 - ASSUMES D1 CONTAINS VALUE DESIRED TO
547             **                         UPDATE FORSTK.
548             ** Exit:
549             **      A(A), D1 point to new top of the FOR/NEXT stack
550             **      Carry clear
551             **
552             ** Calls:      none
553             **
554             ** Stack lvls: 0
555             **
556             ** Uses:     A(A), D1
557             **
558             ** History:
559             **
560             **    Date     Programmer   Modifications
561             **  --------   ----------   -------------
562             **  07/01/82   S.W.         Added documentation
563             **
564             ********************************************************************
565             ********************************************************************
566             *
567             * COLLAPSE ENTRY ON FOR-NEXT STACK
568 0892E 17F   NXTCK7 D1=D1+ 16
569 08931 173          D1=D1+ 4              POSITION TO BOTTOM OF STACK ENTRY
570             * UPDATE POINTER
571 08934 133   NXTCK9 AD1EX
572 08937 1F99          D1=(5) =AVMEME
          5F2
573 0893E 141          DAT1=A A
574 08941 174          D1=D1+ (FORSTK)-(AVMEME)
575 08944 141          DAT1=A A
576 08947 131          D1=A
577 0894A 03           RTNCC
578             *
579             *
580             ********************************************************************
```

```
581             ****************************************************************
582             **
583             ** Name:    XQBLK?  -  Execute Block?
584             **
585             ** Category:   LOCAL
586             **
587             ** Purpose:
588             **      Determines whether a FOR/NEXT loop is to be repeated
589             **      (or executed at all)
590             **
591             ** Entry:
592             **      D1 POINTING AT STEP VALUE ON STACK
593             **      INDEX VALUE IN B(W)
594             **
595             ** Exit:
596             **      P may not be 0 !
597             **      D1 POINTING AT LIMIT VALUE ON STACK
598             **      HEX mode
599             **      CARRY SET => REPEAT (OR INITIATE) LOOP
600             **            CLR => LOOP NOT TO BE REPEATED (OR INITIATED)
601             **
602             ** Calls:      TST12A
603             **
604             ** Uses:       A, B, C, P, D1
605             **
606             ** Stk lvls:   2
607             **
608             ** Algorithm:
609             **   IF (INDEX VBL-LIMIT)*SGN(INCREMENT) > 0 THEN DON'T DO LOOP
610             **   E.G. A STEP SIZE OF ZERO MEANS AN INFINITE LOOP
611             **
612             ** History:
613             **
614             **    Date     Programmer   Modifications
615             **  --------   ----------   ------------
616             **  07/01/82   S.W.         Added documentation
617             **
618             ****************************************************************
619             ****************************************************************
620             *.............................................................*
621             *   Do the (INDEX-LIMIT)*SGN(STEP) =< 0  test. -- J.T.        .
622                                                                           .
623 0894C    XQBLK?                          D1 AT STEP VALUE
624 0894C 1537      A=DAT1 W                STEP IN A(W)
625 08950 17F       D1=D1+ 16               POINT AT LIMIT
626 08953 1577      C=DAT1 W                C=LIMIT                           .
627 08957 AFC       ABEX   W                B=STEP, A=INDEX, C=LIMIT          .
628 0895A 2E        P=     14                                                 .
629 0895C 919       ?B=0   WP               !STEP!=0 ?                        .
630 0895F 71        GOYES  XQEND            execute block (carry set)         .
631 08961 05        SETDEC                                                    .
632 08963 949       ?B=0   S                STEP>0?                           .
633 08966 80        GOYES  FRTEST                                             .
634 08968 BCC       A=-A-1 S                I:=-I                             .
635 0896B BCE       C=-C-1 S                LIM:=-LIM                         .
```

```
636 0896E 23    FRTEST  P=     3              P="<="                          .
637 08970 8E00  utst12  GOSUBL =TST12A        K<=LIMIT ?
          00
638 08976 04    XQEND   SETHEX                                                .
639 08978 01            RTN                   carry set if true, clear otherwise.
640             *
641             *
642             *
643             ***************************************************************
644             ***************************************************************
645             **
646             ** Name:    INDXSN  -  Index Scan
647             **
648             ** Category: LOCAL
649             **
650             ** Purpose:  SCANS FOR-NEXT STACK FOR AN INDEX VARIABLE WHICH
651             **           MATCHES THE ONE IN PROGRAM LINE.
652             **
653             ** Entry:    DO POINTS TO INDEX VARIABLE IN PROGRAM LINE>
654             **           P=0
655             **
656             ** Exit:  B(3-0) CONTAINS INDEX VARIABLE
657             **        P=0
658             **        CARRY SET=> MATCH FOUND.  D1 POINTING TO THE MATCHING
659             **                    INDEX VARIABLE ON THE FOR-NEXT STACK.
660             **                    C(3-0) ALSO CONTAINS INDEX VARIABLE
661             **              CLR=> NO MATCH FOUND.
662             **
663             **        S8=1 => INDEX VARIABLE IN PROGRAM LINE IS ALPHA-DIGIT,
664             **                  OTHERWISE ALPHA
665             **
666             ** Calls:    RANGE
667             **
668             ** Stack lvls: 0
669             **
670             ** Uses:     A(A), B(A), C(A), D1, R0, S8
671             **
672             ** History:
673             **
674             **    Date     Programmer   Modifications
675             **  --------   ----------   -------------
676             **  07/01/82   S.W.         Added documentation
677             **
678             ***************************************************************
679             ***************************************************************
680             *
681 0897A 858    INDXSN ST=1   8             Alpha-digit variable flag
682 0897D 15A3         A=DAT0 4             Read in index var from program
683 08981 8E00         GOSUBL =ARANGE
          00
684 08987 23           P=     3
685 08989 490          GOC    INDX02
686             * Single-character alpha variable
687 0898C 848          ST=0   8
688 0898F D1           B=0    A
```

```
    689 08991 21          P=     1
    690             *
    691 08993 A98   INDX02 B=A    WP
    692 08996 23          P=     3
    693 08998 1FE9        D1=(5) =FORSTK
              5F2
    694 0899F 147         C=DAT1 A
    695 089A2 108         RO=C                  SAVE PTR TO TOP OF FOR-NEXT STACK
    696 089A5 174         D1=D1+ 5
    697 089A8 143         A=DAT1 A              PTR TO 'BOTTOM' OF FOR-NEXT STACK
    698 089AB 135   INDX04 D1=C
    699 089AE 8BA         ?C>=A  A              DONE SEARCHING?
    700 089B1 C1          GOYES  INDX07
    701 089B3 17F         D1=D1+ 16
    702 089B6 17F         D1=D1+ 16
    703 089B9 174         D1=D1+ 5
    704 089BC 147         C=DAT1 A              READ VBL ON STACK
    705 089BF 911         ?B=C   WP             MATCH?
    706 089C2 D0          GOYES  INDX08
    707 089C4 173         D1=D1+ 4
    708 089C7 137         CD1EX
    709 089CA 50E         GONC   INDX04         (B.E.T.)
    710             *
    711 089CD 0D    INDX07 P=P-1                CLEAR CARRY
    712 089CF 20    INDX08 P=     0
    713 089D1 01          RTN
    714             *
    715             *
    716             ********************************************************************
    717             ********************************************************************
    718             **
    719             ** Name:    POPSH   -  Pop/Push
    720             **
    721             ** Category: LOCAL
    722             **
    723             ** Purpose:  Calls EXPEX- (1st collapsing any residual data on
    724             **           the expression stack), ensures correct data type,
    725             **           then pushes the value on the FOR/NEXT stack.
    726             **
    727             ** Entry:    D0 must point 2 nibs prior to start of expression
    728             **
    729             ** Exit:     D0 past expression.
    730             **           D1 reflects the new top of the FOR/NEXT stack.
    731             **           P= 0
    732             **           Error exits if expression isn't real numeric.
    733             **           A(W) contains the resultant numeric value.
    734             **
    735             ** Calls:    EXPEX-, POP1R, C=MSTK
    736             **
    737             ** Uses:     A-D, R0-R3, D1,D0, S0-S11, all of function scratch
    738             **
    739             ** Stack lvls: 5
    740             **
    741             ** History:
    742             **
```

```
743                **    Date     Programmer   Modifications
744                **  --------   ----------   -------------
745                **  07/01/82   S.W.         Added documentation
746                **
747                ********************************************************************
748                ********************************************************************
749                *
750 089D3 161    POPSH  D0=D0+ 2               SKIP OVER TOKEN
751 089D6 8E00         GOSUBL =EXPEX-          THROW EXP ON ARITH. STACK
           00
752 089DC 8E00         GOSUBL =POP1R           TAKE FORSTK AS AVMEME
           00
753 089E2 04           SETHEX
754                *
755 089E4 7800         GOSUB  C=MSTK
756                *
757 089E8 145    DT1=C  DAT1=C A               UPDATE FORSTK
758 089EB 135          D1=C
759 089EE 03           RTNCC
760                *
761                * Leaves D1 at FORSTK and C(A) 16 nibs above the top of the
762                * FOR-NEXT stack
763                *
764 089F0 8F00   C=MSTK GOSBVL =D1FSTK
           000
765 089F7 1CF          D1=D1- 16
766 089FA 137          CD1EX
767 089FD 03           RTNCC
768                *
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:    LINSKP  -  Line Skip
                ** Name:(S) LNSKP-  -  Line Skip
                **
                ** Category:   EXCUTL
                **
                ** Purpose:    Skips to next statement
                **
                ** Entry:      2 entry points:
                **               1) LNSKP-  -  PCADDR points to stmt length byte
                **               2) LINSKP  -  D0 points to stmt length byte
                **
                ** Exit:     D0 points to end of statement token (t@ or tEOL)
                **           A(A) = D0
                **           B(B) = Statement length
                **           Carry Clear
                **
                ** Calls:      D0=PCA (LNSKP- entry only)
                **
                ** Stack lvls: 1 (LNSKP- only)
                **             0 (LNSKP entry)
                **
                ** Uses:     A(A), B(A), D0
                **
                ** History:
                **
                **    Date     Programmer   Modifications
                **  --------   ----------   -------------
                **  07/01/82   S.W.         Added documentation
                **  10/15/82   S.W.         Call to D0=PCA to save code
                **
                ********************************************************************
                ********************************************************************
                * SCANS TO NEXT LINE OF PROGRAM MEMORY
                *
089FF 8E00      =LNSKP- GOSUBL =D0=PCA
      00
                * D0 POINTS AT L.L. BYTE
08A05 D0        =LINSKP A=0    A
08A07 14A               A=DAT0 B               LINE LENGTH
08A0A D8                B=A    A
08A0C 132               ADOEX
08A0F C0                A=A+B  A               POINTS TO EOL OR @
08A11 130               D0=A
08A14 03         rtncc  RTNCC
```

```
816                     STITLE NXTSTM - Next statement scan
817             ****************************************************************
818             ****************************************************************
819             **
820             ** Name:(S) NXTSTM  -  Scan to Next Stmt/Jump to BASIC Loop
821             **
822             ** Category: EXCUTL
823             **
824             ** Purpose:  Next statement scan & jump to BASIC loop @ RUNRTN
825             **
826             ** Entry:    ENTRY POINTS:
827             **             NXTSTM - entry point to go on to the following
828             **                      statement.  No assumptions made.
829             **                      PCADDR must be current.
830             **                      sENDx flag will be explicitly cleared.
831             **                      entry point for IMAGE & REM.
832             **             NXTST1 - Entry point for END execute. (sENDx=1)
833             **                      PCADDR must be current.
834             **             NXTST2 - DO points at statement length byte.
835             **                      Assumes sENDx is clear
836             **             NXTST3 - DO points at EOL token
837             **                      Assumes sENDx is clear
838             **             NXTST5 - DO already points at EOL token
839             **                      Explicitly clears sENDx
840             **                      Entry pt for routines which may
841             **                      have inadvertantly set sENDx, perhaps
842             **                      via EXPEXC
843             **
844             **             LABEL  - Label 'execute' (NOP)
845             **             DATA   - DATA statement execute (NOP)
846             **             BANG   - REM (!) execute (NOP)
847             **
848             ** Exit:     DO POINTS TO @ OR EOL TOKEN
849             **           Through RUNRTN
850             **
851             **           LABEL:
852             **             Skips ASCII Label
853             **             If Multi-statement line   ("@")
854             **                Through RUNXLP (to avoid SST between Labels)
855             **             else
856             **                Through RUNRTN (with DO @ EOL)
857             **
858             ** Calls:      none
859             **
860             ** Stack lvls: 0
861             **
862             ** Uses:     A(A), B(A), C(A), DO, S1 (sENDx)
863             **
864             ** Detail:   USED TO 'EXECUTE' REM, LABEL, DATA STATEMENTS
865             **
866             **           The END Execute flag is ALWAYS cleared by NXTSTM
867             **           END enters at NXTST1 with sENDx set
868             **           This is necessary when a program is NOT to continue
869             **
870             ** Label Execute:
```

```
                **   @EOL return to BASIC loop
                **
                ** History:
                **
                **    Date     Programmer   Modifications
                **  --------   ----------   ------------
                **  07/01/82   S.W.         Added documentation
                **  03/30/83   J.P.         Shift C(B) for ASCII check
                **
                ****************************************************************
                ****************************************************************
                * SCANS TO NEXT LINE OF PROGRAM MEMORY
                *
08A16 00000             REL(5) =DATADC
      0
08A1B 00000             REL(5) =DATP
      0
08A20 6720  =DATA   GOTO   NXTSTM
                *
                * To prevent SST from stopping between Labels:
                *    Check if End of LABEL statement = EOL
                *      High bit set --.> @ (F4)  and EOL (F0)
                *    If not, enter BASIC Loop to avoid SST pause
                *
08A24 162   =LABEL  DO=DO+ 3                Skip 3 nibs of Jump address
08A27 161    LABL10 DO=DO+ 2
08A2A 14E           C=DATO B                Read ASCII char of Label
08A2D A66           C=C+C  B                Check for High bit set
08A30 56F           GONC   LABL10           Still ASCII
08A33 90A           ?C=0   P                EOL ?
08A36 F2            GOYES  NXTST3           goto RUNRTN- assumes sENDx=0
08A38 8C00          GOLONG =BSCXLP          Multi-statement Label
      00
08A3E 00000             REL(5) =REMDC
      0
08A43 00000             REL(5) =REMP
      0
08A48       =IMAGE
08A48       =REM
08A48       =BANG
08A48 841   =NXTSTM ST=0   sENDx            Clear END Execute flag
08A4B 1B97  =NXTST1 DO=(5) =PCADDR          END statement Entry
      6F2
08A52 142           A=DATO A
08A55 130           DO=A
                * DO POINTS AT L.L. BYTE
                *  To save cycle time, don't call LINSKP
08A58 D0    =NXTST2 A=0    A
08A5A 14A           A=DATO B                Read in stmt length
08A5D 136           CDOEX
08A60 CA            A=A+C  A                Position to stmt terminator
08A62 130           DO=A
08A65 8C00   NXTST3 GOLONG =RUNRTN          Return to Run Loop
      00
                *
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) TKSCN+  -  Token Scan
                ** Name:    TKSCN4  -  Token Scan
                ** Name:(S) TKSCN7  -  Token Scan
                **
                ** Category:   EXCUTL
                **
                ** Purpose:    Search program memory (or statement buffer) for
                **             a specific 2 nibble begin BASIC token
                **
                ** Entry:      C(B) contains token to match on
                **             P=0
                **             D(A)= PRGMEN if in a program
                **                 = end of statement buffer, otherwise
                **             3 Entry points:
                **               1) TKSCN+ - DO at tEOL before search start
                **               2) TKSCN4 - DO at some statement length byte
                **               3) TKSCN7 - DO at tEOL or t@ before search start
                **
                ** Exit:     CARRY SET => Token found & DO points to it.
                **           CARRY CLR => Searched to program end
                **                        (or statement buffer end) without
                **                        finding a match.
                **
                **
                ** Calls:      none
                **
                ** Stack lvls: 0
                **
                ** Uses:     A(A),B(A),C(A),DO
                **
                ** History:
                **
                **    Date     Programmer   Modifications
                **  --------   ----------   -------------
                **  07/01/82   S.W.         Added documentation
                **  07/07/82   S.W.         All references to F-RO-0 & S9
                **                           to save CURRL have been
                **                           eliminated.
                **
                ****************************************************************
                ****************************************************************
                *
08A6B 161   =TKSCN+ DO=DO+ 2              POINTS TO LINE NUMBER
08A6E 163    TKSCN2 DO=DO+ 4              POINT TO LINE LENGTH
08A71 136   =TKSCN4 CDOEX
08A74 8BF           ?C>=D  A               NO MORE PGM MEM TO SEARCH?
08A77 D9            GOYES  rtncc
08A79 136           CDOEX
08A7C DO            A=0    A
08A7E 14A           A=DAT0 B               LINE LENGTH
08A81 D8            B=A    A
```

```
974 08A83 161          D0=D0+ 2              PT TO BEGIN BASIC TOKEN
975 08A86 14A          A=DAT0 B              TOKEN IN PGM MEMORY
976 08A89 962          ?A=C   B              SAME AS SEARCH TOKEN?
977 08A8C 00           RTNYES                CARRY=>MATCHING TOKEN FOUND
978 08A8E 132          ADOEX                 A(A) pts to begin BASIC token
979 08A91 C0           A=A+B  A              A(A) pts past @ or EOL
980 08A93 132          ADOEX
981 08A96 181          D0=D0- 2              BACK UP TO @ OR EOL
982 08A99 14A  =TKSCN7 A=DAT0 B
983 08A9C 161          D0=D0+ 2              Pt at line# OR stmt length
984             *                             (preceded by tEOL or t@)
985 08A9F 90C          ?A#0   P              NOT EOL?
986 08AA2 FC           GOYES  TKSCN4
987             * POINTING AT LINE#
988             *
989 08AA4 59C          GONC   TKSCN2         (B.E.T.)
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) EOLSCN  -  tEOL Scan
                ** Name:    EOLSN5  -  tEOL Scan
                ** Name:    EOLSN7  -  tEOL Scan
                **
                ** Category:   EXCUTL
                **
                ** Purpose:    Scans to tEOL (as opposed to t@ OR tEOL)
                **
                **             3 entry points:
                **               1) EOLSCN -  PCADDR at current stmt len byte
                **               2) EOLSN5 -  D0 at t@ or tEOL
                **                            C(B)=tEOL
                **               2) EOLSN7 -  D0 at t@
                **                            C(B)=tEOL
                **
                ** Exit:       D0 POINTS TO EOL; A(B) = EOL TOKEN;  CARRY SET
                **             If EOLSCN entry point used, P=0.
                **
                ** Calls:      LINSKP
                **
                ** Uses:       A(A), B(A), C(B), D0
                **
                ** Stack lvls: 2
                **
                ** History:
                **
                **    Date      Programmer   Modifications
                **  --------    ----------   -------------
                **  07/01/82    S.W.         Added documentation
                **  01/17/83    S.W.         Added EOLSN5 entry point
                **
                ********************************************************************
                ********************************************************************
                *
08AA7 20        =EOLSCN P=      0
08AA9 310F              LCHEX   F0
08AAD 7E4F              GOSUB   LNSKP-
08AB1 14A       =EOLSN5 A=DAT0  B               READ IN STATEMENT TERMINATOR
08AB4 962               ?A=C    B               IS IT EOL?
08AB7 00                RTNYES
08AB9 161       =EOLSN7 D0=D0+  2
08ABC 754F              GOSUB   LINSKP
08AC0 50F               GONC    EOLSN5          (B.E.T.)
```

```
1037                    EJECT
1038                ********************************************************************
1039                ********************************************************************
1040                **
1041                ** Name:    KEY     -  Executes KEY statement
1042                **
1043                ** Category:   STEXEC
1044                **
1045                ** Purpose:    Executes KEY statement
1046                **
1047                ** Entry:      DO is past tKEY (pointing to string expression)
1048                **
1049                ** Exit:       Key assignment (or de-assignment) is reflected
1050                **             in keys file.
1051                **             Exits via NXTSTM
1052                **
1053                ** Calls:      GYKYCD, KEYDEL, EXPEXC, REV$, KEYMRG, MOVEUM
1054                **
1055                ** Uses:
1056                **  exclusive... A-C, P, R0-R3, S8, S-R0-0, F-R0-1
1057                **  inclusive... A-D, all of function scratch, S0-S11 - EXPEXC
1058                **
1059                ** Detail:   KEY ASSIGNMENT STORED AS: dd ll t string literal
1060                **           WHERE dd IS THE HEX KEYCODE, ll IS A 2 NIBBLE HEX
1061                **           REPRESENTATION OF THE ENTIRE LENGTH OF THE KEY
1062                **           ASSIGNMENT ENTRY, & t IS A ONE NIBBLE ENCODING OF
1063                **           THE ASSIGNMENT TYPE: 0 (terminating),
1064                **           1 (; - non-terminating), 2 (: - immediate execute).
1065                **           THE STRING EXPRESSION TO BE ASSIGNED IS EXECUTED
1066                **           PRIOR TO BEING STORED IN THE KEY ASSIGNMENT AREA.
1067                **           KEY ASSIGNMENTS STORED IN ROW MAJOR ORDER.
1068                **
1069                ** Stack lvls: 6
1070                **
1071                ** History:
1072                **
1073                **    Date     Programmer   Modifications
1074                **   --------   ----------   ------------
1075                **   10/03/82   S.C.         Replace GOSUB PCCHK by ?S13=0
1076                **                            assuming S13 is valid
1077                **   07/01/82   S.W.         Added documentation
1078                **
1079                ********************************************************************
1080                ********************************************************************
1081                *
1082                *
1083                *
1084 08AC3 0000         REL(5) =KEYDC
            0
1085 08AC8 0000         REL(5) =KEYP
            0
1086 08ACD 71C2 =KEY    GOSUB  GTKYCD
1087                * KEYCODE IN ROW MAJOR ORDER
1088                * FOUND VALID KEYCODE
1089                *
```

```
1090 08AD1 14A          A=DAT0 B
1091 08AD4 3100         LC(2)  =tCOMMA
1092 08AD8 962          ?A=C   B
1093 08ADB 11           GOYES  KEY42
1094             *
1095 08ADD 7B42         GOSUB  KEYDEL
1096 08AE1 666F         GOTO   NXTSTM
1097             *
1098 08AE5 8D00  rdatty GOVLNG =RDATTY       DATA TYPE
           000
1099             *
1100 08AEC AE9   KEY42  C=B    B
1101 08AEF 1F17         D1=(5) =S-R0-0
           8F2
1102 08AF6 14D          DAT1=C B             SAVE KEYCODE
1103             *
1104 08AF9 161          D0=D0+ 2             STEP OVER COMMA TOKEN
1105 08AFC 8E00         GOSUBL =EXPEXC
           00
1106 08B02 31F0         LCHEX  0F
1107 08B06 966          ?A#C   B             NOT STRING?
1108 08B09 CD           GOYES  rdatty
1109 08B0B 8F00         GOSBVL =REV$         REVERSE STRING ON STACK
           000
1110 08B12 171          D1=D1+ 2
1111 08B15 143          A=DAT1 A
1112 08B18 D8           B=A    A             STRING LENGTH
1113 08B1A D2           C=0    A
1114 08B1C 31AF         LCHEX  FA            MAXIMUM #CHARS IS 125 (250 NIBS)
1115 08B20 8BA          ?A<=C  A
1116 08B23 40           GOYES  KEY45
1117 08B25 D5           B=C    A             FF IS MAX KEY ASSIGN ENTRY LENGTH
1118 08B27 17D   KEY45  D1=D1+ 14            POSITION D1 AT STRING ON STACK
1119 08B2A 14A          A=DAT0 B
1120 08B2D AC1          B=0    S
1121 08B30 31B3         LCASC  \;\
1122 08B34 962          ?A=C   B
1123 08B37 D0           GOYES  KEY47
1124 08B39 30A          LC(1)  \:\
1125 08B3C 966          ?A#C   B             NOT ':'?
1126 08B3F 80           GOYES  KEY50
1127 08B41 B45          B=B+1  S
1128 08B44 B45   KEY47  B=B+1  S             B(S)= 0=>stmt end  1=>;  2=>:
1129             *
1130 08B47 1B17  KEY50  D0=(5) =S-R0-0
           8F2
1131 08B4E D0           A=0    A
1132 08B50 14A          A=DAT0 B
1133             *
1134 08B53 DC           ABEX   A             RESTORE KEYCODE
1135 08B55 7630         GOSUB  KEYMRG
1136             *
1137             * WRITE OUT NEW ENTRY
1138 08B59 112          A=R2
1139 08B5C 1593         DAT1=A 4             WRITE OUT KEYCODE & LENGTH
```

```
1140 08B60 173          D1=D1+ 4
1141 08B63 1514         DAT1=A S              WRITE OUT ENCODING FOR ; OR :
1142 08B67 170          D1=D1+ 1              BGN DEST.
1143            *
1144            * NOW PREPARE TO COPY UP STRING ASSIGNMENT
1145 08B6A 1B99         DO=(5) =AVMEME
          5F2
1146 08B71 146          C=DAT0 A
1147 08B74 134          DO=C                  AVMEME
1148 08B77 BF4          ASR    W
1149 08B7A BF4          ASR    W              LENGTH OF ENTRY
1150 08B7D D2           C=0    A
1151 08B7F 305          LCHEX  5
1152 08B82 EE           C=A-C  A              STRING LENGTH
1153 08B84 16F          DO=DO+ 16             BGN SOURCE
1154            *
1155 08B87 73B7 =MOVNXT GOSUB  MOveu3         COPY STRING
1156 08B8B 6CBE         GOTO   NXTSTM
1157            *
1158            *
1159            ************************************************************************
1160            ************************************************************************
1161            **
1162            ** Name:(S) KEYMRG  -  Key Merge
1163            ** Name:    KYMRG+  -  Key Merge
1164            **
1165            ** Category:   FILUTL
1166            **
1167            ** Purpose:    Creates space for new entry in keys file
1168            **
1169            ** Entry:    P=     0
1170            **           B(A) = HEX Keycode
1171            **           2 ENTRY POINTS:
1172            **             1) KEYMRG - A(A) =Length of assignment string
1173            **             2) KYMRG+ - C(B) =Keycode
1174            **                         C(6-2)= Length of assignment string
1175            **
1176            ** Exit:     D1 points to start of new entry
1177            **           R2(B) = Keycode; R2(3-2) = Entry length
1178            **           R2(S) = B(S) on entry
1179            **           B(A)  = offset to memory
1180            **           R3    = Pointer to keys file header
1181            **           Carry clear
1182            **           via RSTD1
1183            **
1184            ** Calls:    KMEMCK, CREATF, MOVEDM, RFADJ+, KYD30,
1185            **           KEYFND, KYPRCK, LAKEYS, UPDFCL
1186            **
1187            ** Uses:     A-D, D1, DO, R0-R3, F-R0-1, S6, S8
1188            **
1189            ** Stack lvls: 5
1190            **
1191            ** History:
1192            **
1193            **    Date     Programmer   Modifications
```

```
1194                ** --------  ----------  -------------
1195                ** 07/01/82  S.W.        Added documentation
1196                ** 11/02/82  S.W.        Added call to UPDFCL
1197                ** 12/29/82  S.W.        Eliminated call to RFAD85
1198                **
1199                ************************************************************
1200                ************************************************************
1201                *
1202 08B8F D2     =KEYMRG C=0    A
1203 08B91 305         LCHEX  5
1204 08B94 C2          C=C+A  A              TOTAL ENTRY LENGTH
1205 08B96 BF2         CSL    W
1206 08B99 BF2         CSL    W
1207 08B9C AE9         C=B    B
1208 08B9F AC9         C=B    S              C(S)=term;C(B)=KC;C(3-2)=Entry len.
1209 08BA2 10A  =KYMRG+ R2=C
1210                *
1211 08BA5 7F01        GOSUB  KEYFND
1212 08BA9 112         A=R2
1213 08BAC AF8         B=A    W
1214 08BAF BF5         BSR    W
1215 08BB2 BF5         BSR    W              B(A)=LENGTH OF NEW ENTRY
1216 08BB5 4E3         GOC    KMRG30
1217 08BB8 D2          C=0    A
1218 08BBA 868         ?ST=0  8              keys FILE EXISTS?
1219 08BBD 73          GOYES  KMRG30
1220                * CREATE keys FILE
1221 08BBF 3152        LC(2)  (oFLENh)+(oKYsod)
1222 08BC3 C1          B=B+C  A              ENTIRE AMT OF MEM NEEDED TO ADD
1223 08BC5 7280        GOSUB  KMEMCK
1224                *
1225 08BC9 D9          C=B    A              SIZE OF FILE TO CREATE
1226 08BCB 7000        GOSUB  =CREATF        CREATE FILE
1227 08BCF 119         C=R1
1228 08BD2 135         D1=C                  PTR TO FILE HEADER
1229 08BD5 8E00        GOSUBL =LAKEYS
          00
1230 08BDB 15DF        DAT1=C 1FNAMh         WRITE OUT FILE NAME
1231 08BDF 17F         D1=D1+ oFTYPh
1232 08BE2 33C0        LC(4)  =fKEY
          2E
1233 08BE8 15D3        DAT1=C 1FTYPh         WRITE OUT FILE TYPE
1234 08BEC 17F         D1=D1+ (1FLAGh)+(1FTYPh)+(1DATEh)+(1TIMEh)
1235 08BEF 174         D1=D1+ (1FLENh)       POINT TO HEADER END
1236 08BF2 01          RTN
1237                *
1238                * KEY FOUND - IF R2-C(A)>0, GO ON TO KMRG50 (EXPAND ENTRY)
1239                * ELSE CONTRACT ENTRY
1240                *
1241 08BF4 133   KMRG30 AD1EX                 SAVE PTR TO EXISTING ENTRY
1242 08BF7 1F0A        D1=(5) =F-R0-1
          8F2
1243 08BFE 141         DAT1=A A
1244 08C01 7951        GOSUB  KYPRCK         CK PROTECTION OF keys FILE
1245 08C05 143         A=DAT1 A
```

```
1246 08C08 CA          A=A+C  A              END OF EXISTING ENTRY (BGN SOURCE)
1247 08C0A 100         R0=A                 SAVE BGN SOURCE
1248 08C0D E1          B=B-C  A
1249 08C0F 501         GONC   KMRG55
1250           * CONTRACT BY AMT. IN B; D1 IS BGN SOURCE; D1+B IS BGN DEST.
1251 08C12 D6          C=A    A
1252 08C14 C9          C=C+B  A              BGN DESTINATION
1253 08C16 135         D1=C
1254 08C19 7431        GOSUB  KYD30         Moves mem,calls RFADJ-,updates ptrs
1255 08C1D 5D1         GONC   KMRG85        (B.E.T.)
1256           *
1257           * EXPAND keys FILE BY AMT. IN B - A, R0 POINT TO BGN SOURCE
1258 08C20 7720  KMRG55 GOSUB  KMEMCK
1259 08C24 C0          A=A+B  A              END DEST.
1260 08C26 131         D1=A
1261 08C29 E0          A=A-B  A              End source
1262 08C2B 118         C=R0                 Bgn source
1263 08C2E 8F00        GOSBVL =MOVED2
         000
1264           * Begin Source is in R0
1265 08C35 8E00        GOSUBL =RFAD++
         00
1266           *
1267 08C3B 11B   KMRG85 C=R3                 PTR TO HEADER START
1268 08C3E 8E00        GOSUBL =UPDFCL
         00
1269           * WANT D1 POSITIONED AT ENTRY STILL
1270 08C44 8D00 =rstd1 GOVLNG =RSTD1
         000
1271           *
1272           *
1273           ******************************************************************
1274           ******************************************************************
1275           **
1276           ** Name:    KMEMCK  -  Memory Check
1277           **
1278           ** Category:   GENUTL
1279           **
1280           ** Purpose:  Checks amount of memory left
1281           **
1282           ** Entry:    B(A)= Amount of memory needed
1283           **           P=0 to ensure check with leeway
1284           **
1285           ** Exit:     RETURN => THERE'S ENOUGH MEMORY
1286           **                     Carry set
1287           **                     D1 at AVMEMS
1288           **                     B(A) preserved
1289           **                     A(A) = (AVMEMS)
1290           **                     C(A) = memory left after allocation
1291           **             ELSE ERROR EXITS
1292           **
1293           ** Calls:    MEMCL+
1294           **
1295           ** Stack lvis: 1
1296           **
```

```
1297            ** Uses:     A(A), C(A), D1
1298            **
1299            ** History:
1300            **
1301            **    Date     Programmer   Modifications
1302            **  --------   ----------   -------------
1303            **  07/01/82   S.W.         Added documentation
1304            **  10/12/82   S.W.         Calls MEMCL+ to check with
1305            **                           leeway.
1306            **
1307            ************************************************************************
1308            ************************************************************************
1309            *
1310            *
1311 08C4B 8E00 =KMEMCK GOSUBL =MEMCL+
          00
1312 08C51 500          RTNNC
1313            *
1314 08C54 6000  mferr  GOTO   =MFERR
1315            *
1316            *
1317 08C58 136   trcfrm CDOEX
1318 08C5B 10A          R2=C                    Save D0
1319 08C5E 8E00         GOSUBL =TRFROM
          00
1320 08C64 11A   trcfr1 C=R2                    Restore D0
1321 08C67 134          D0=C
1322 08C6A 01           RTN
1323 08C6C 8C00  trflck GOLONG =TRFLCK
          00
1324            *
1325 08C72 76FF =SNcrlf GOSUB  trflck
1326 08C76 400          RTNC
1327 08C79 8C00         GOLONG =CRLFSD
          00
1328            *
```

```
1329                    STITLE PSHSTK - Open up Stack
1330            ********************************************************************
1331            ********************************************************************
1332            **
1333            ** Name:(S) PSHSTK  -  Push Stack
1334            ** Name:(S) PSHSTL  -  Push Stack
1335            **
1336            ** Category:   GENUTL
1337            **
1338            ** Purpose:    Moves high memory to lower memory to allow 'push'
1339            **             onto GOSUB, VARIABLE, or some other stack.
1340            **
1341            **             Push address on stack with NO LEEWAY check
1342            **
1343            ** Entry:
1344            **           D0     pointer to top of stack pointer
1345            **           B(A)=  Amt memory needs to 'open up'.
1346            **           PSHSTK:
1347            **             P=n-1  where n=# pointers to be adjusted
1348            **             LEEWAY will ALWAYS be checked
1349            **           PSHSTL:
1350            **             C(0) = # pointers to be adjusted
1351            **             P= non-zero if LEEWAY not to be checked
1352            **
1353            ** Exit:     Carry Clear:
1354            **             B(A) is preserved
1355            **             P=0
1356            **             D1 points to new top of stack
1357            **             RAM pointers are adjusted
1358            **           Error Exit
1359            **             Insufficient Memory to open stack
1360            **
1361            ** Calls:    MOVEU1, PTRAD1, MEMCL+
1362            **
1363            ** Uses:     A, C(A), D(A), D0, D1
1364            **
1365            ** Detail:   Usefulness of this routine could be extended
1366            **           to variable creation, CALL/SUB. etc
1367            **
1368            **           GOSUB required C(S) not be altered.
1369            **
1370            **           Preserves math stack.
1371            **
1372            ** Stack lvls: 1
1373            **
1374            ** History:
1375            **
1376            **    Date     Programmer   Modifications
1377            **  --------   ----------   -------------
1378            **  07/04/82   S.W.         Added documentation
1379            **  08/10/82   S.W.         Modified to preserve math stk
1380            **  09/30/82   J.P.         Added MEMCL+ call, removed R1
1381            **  10/12/82   S.W.         Changed D=C B to A field.
1382            **                          Replaced MEMCL+ with KMEMCK.
1383            **  10/29/82   S.W.         Took out KMEMCK call, due to
```

```
1384                **                              subroutine levels - PSHSTK
1385                **                              to be used by GOSUB/GOSUB
1386                **  02/15/83   J.P.            Added PSHSTL entry for no
1387                **                              LEEWAY check
1388                **
1389                ********************************************************************
1390                ********************************************************************
1391                *
1392                *
1393 08C7F 80F0 =PSHSTK CPEX   0
1394 08C83 20          P=     0                     Mem Check with Leeway setting
1395                *
1396                * Entry point so LEEWAY will not be checked
1397                *   P should be non zero
1398                *   C(0) = # pointers to adjust
1399                *
1400 08C85 D7   =PSHSTL D=C    A                     SAVE #PTRS TO ADJUST
1401 08C87 8E00        GOSUBL =MEMCL+               Enough memory ?
           00
1402                * D1 at AVMEMS
1403 08C8D 46C         GOC    nferr
1404                *
1405                * For entry to MOVEU1
1406                *   D0 @ Pointer to End Source
1407                *   D1 @ Start of Destination
1408                *    A @ Start of source
1409                *
1410 08C90 1D99        D1=(2) =AVMEME
1411 08C94 143         A=DAT1 A                     Start of Source
1412 08C97 D6          C=A    A
1413 08C99 E9          C=C-B  A                     Start of DESTINATION
1414 08C9B 135         D1=C
1415 08C9E 8F00        GOSBVL =MOVEU1               CREATE SPACE
           000
1416                * D0= END SOURCE (END DEST. FOR COPY DOWN)
1417                * D1= END DEST.  ( NEW TOP OF STACK)
1418                *
1419 08CA5 F9          B=-B   A                     POINTERS WILL BE DECREMENTED
1420 08CA7 136         CDOEX
1421                *
1422                * Removed R1=C setting
1423                *   Cannot used "R" registers, no one calling PSHSTK used R1
1424                *
1425 08CAA DB          C=D    A
1426 08CAC 80F0        CPEX   0                     RESTORE P
1427 08CB0 7FC2        GOSUB  PTRAD1                ADJUST PTRS
1428 08CB4 F9          B=-B   A                     AMT. OF NIBBLES TO PUSH ON STACK
1429 08CB6 03          RTNCC
1430                *
1431                *
1432                ********************************************************************
1433                ********************************************************************
1434                **
1435                ** Name:(S) KEYFND  -  Key Assignment Find
1436                ** Name:    KYFND+  -  Key Assignment Find
```

```
1437              **
1438              ** Category: EXCUTL
1439              **
1440              ** Purpose:  FINDS SPECIFIED KEY ASSIGNMENT IN keys FILE
1441              **
1442              ** Entry:    P=   0
1443              **      2 entry points:
1444              **             1) KEYFND - B(A)=keycode
1445              **             2) KYFND+ - D(A)=keycode
1446              **                         A(A) points to header of keys file
1447              **
1448              ** Exit:     CARRY CLR=> NO MATCH
1449              **                       D1 points past last entry which
1450              **                       had a smaller keycode value
1451              **                  SET=> MATCH FOUND. D1 AT ENTRY.
1452              **                       C(A)=Entire entry length
1453              **                       D0 points to file header end
1454              **           P=0
1455              **           B(A)=KEYCODE
1456              **           If entry point KEYFND was used then:
1457              **             S8=1=> NO keys FILE
1458              **               =0=> D0 POINTS TO FILE HEADER END
1459              **                    R3 POINTS TO FILE START
1460              **
1461              ** Calls:    FILEF, LAKEYS - only KEYFND entry point
1462              **
1463              ** Uses:
1464              **  exclusive... A, B(A), C, D, D1, D0
1465              **  inclusive... A, B(A), C, D, D1, D0, S6,S8, R3 - KEYFND
1466              **
1467              ** Stack lvls: 1  KEYFND entry
1468              **             0  KYFND+ entry
1469              **
1470              ** History:
1471              **
1472              **    Date     Programmer   Modifications
1473              **  --------   ----------   -------------
1474              **  07/01/82   S.W.         Added documentation
1475              **
1476              ********************************************************************
1477              ********************************************************************
1478              *
1479 08CB8 8E00  =KEYFND GOSUBL =LAKEYS
          00
1480 08CBE DD            BCEX   A
1481 08CC0 D7            D=C    A               SAVE KEY CODE
1482 08CC2 8E00          GOSUBL =FILEF          SEARCH MAINFRAME ONLY
          00
1483 08CC8 4D5           GOC    KYFD30          FILE DOESN'T EXIST?
1484              *
1485 08CCB 848           ST=0   8
1486              * FILE EXISTS - NOW LOOK FOR SPECIFIED KEYCODE
1487 08CCE 133           AD1EX
1488 08CD1 103           R3=A
1489 08CD4 D2   =KYFND+  C=0    A
```

```
1490 08CD6 3102          LC(2)  (oFLENh)
1491 08CDA CA            A=A+C  A              PTR TO FILE LENGTH FIELD
1492 08CDC 131           D1=A                  PTR TO HEADER END
1493 08CDF 3150          LC(2)  =oKYsod
1494 08CE3 C2            C=C+A  A              PTR TO HEADER END
1495 08CE5 134           DO=C
1496 08CE8 143           A=DAT1 A              FILE LENGTH
1497 08CEB 137           CD1EX
1498 08CEE C2            C=C+A  A              Pointer to file end
1499 08CF0 DF            CDEX   A              D(A)=file end ptr; C(A)=keycode
1500 08CF2 D5            B=C    A
1501              *
1502 08CF4 136           CDOEX
1503 08CF7 134           DO=C
1504 08CFA 135   KYFD10  D1=C
1505 08CFD 8BF           ?C>=D  A
1506 08D00 A2            GOYES  KYFD40
1507 08D02 D0            A=0    A
1508 08D04 14B           A=DAT1 B              KEYCODE IN A(A)
1509 08D07 D2            C=0    A
1510 08D09 171           D1=D1+ 2
1511 08D0C 14F           C=DAT1 B              ENTRY LENGTH IN C(A)
1512 08D0F 1C1           D1=D1- 2
1513 08D12 8B8           ?A>=B  A              MUST do test on A field
1514 08D15 A0            GOYES  KYFD20
1515 08D17 133           AD1EX
1516 08D1A C2            C=A+C  A
1517 08D1C 5DD           GONC   KYFD10         (B.E.T.)
1518              *
1519 08D1F 8A0   KYFD20  ?A=B   A
1520 08D22 00            RTNYES
1521 08D24 01            RTN
1522              *
1523 08D26 DB    KYFD30  C=D    A
1524 08D28 D5            B=C    A
1525 08D2A 03    KYFD40  RTNCC                 SA WAS HERE 6-9-82
1526              *
1527              *
1528              ******************************************************************
1529              ******************************************************************
1530              **
1531              ** Name:(S) KEYDEL  -  Key Assignment Delete
1532              **
1533              ** Category:   EXCUTL
1534              **
1535              ** Purpose:  If there's an assignment string associated with
1536              **           specified key, delete it.
1537              **
1538              ** Entry:    B(A) = Keycode
1539              **           P=0
1540              **
1541              ** Exit:     P=0
1542              **           Carry Clear
1543              **           Any assignment to that key is deleted
1544              **             via RFAD--
```

```
1545                **
1546                ** Calls:    KEYFND, MOVEUM, KYPRCK
1547                **
1548                ** Uses:     A-D, D1, D0, R0,R1,R3, S6,S8
1549                **
1550                ** Stack lvls: 3
1551                **
1552                ** History:
1553                **
1554                **    Date     Programmer   Modifications
1555                **  --------   ----------   -------------
1556                **  07/01/82   S.W.         Added documentation
1557                **  12/29/82   S.W.         Eliminated call to RFAD94
1558                **
1559                ****************************************************************
1560                ****************************************************************
1561                *
1562 08D2C 788F =KEYDEL GOSUB  KEYFND
1563 08D30 500         RTNNC
1564 08D33 7720        GOSUB  KYPRCK          CHECK PROTECTION ON keys FILE
1565 08D37 184         D0=D0- oKYsod          POINT TO FILE LENGTH
1566 08D3A 142         A=DAT0 A
1567 08D3D EA          A=A-C  A               NEW FILE LENGTH
1568 08D3F 140         DAT0=A A               UPDATE LENGTH FIELD
1569 08D42 D5          B=C    A
1570 08D44 F9          B=-B   A               OFFSET
1571 08D46 133         AD1EX
1572 08D49 131         D1=A                   BGN DEST.
1573 08D4C CA          A=A+C  A               BGN SOURCE
1574 08D4E 100         R0=A                   SAVE FOR RFADJ-
1575 08D51 8F00  KYD30 GOSBVL =MOVEUA         DELETE ENTRY IN key FILE
          000
1576                *
1577                * Begin source in R0
1578 08D58 8C00        GOLONG =RFAD--
          00
1579                *
1580                ****************************************************************
1581                ****************************************************************
1582                **
1583                ** Name:    KYPRCK  -  Check keys File Protection
1584                **
1585                ** Category:   FILUTL
1586                **
1587                ** Purpose:
1588                **      Checks protection of file pointed to by R3 (assumes
1589                **      KEYFND called) while preserving C(A) & D1
1590                **
1591                ** Entry:
1592                **      P      =  0
1593                **      R3 points to file header
1594                **
1595                ** Exit:
1596                **      P=0
1597                **      R3 points to file header
```

```
1598            **      SB=0
1599            **      Carry Set => File was not secure
1600            **              else error exit
1601            **
1602            ** Calls:      GETPRO
1603            **
1604            ** Uses.......
1605            **  Inclusive: A(A), D(A), SB
1606            **
1607            ** Stk lvls:   2
1608            **
1609            ** History:
1610            **
1611            **    Date     Programmer   Modification
1612            **  --------   ----------   ------------
1613            **  07/01/82   S.W.         Added documentation
1614            **
1615            ************************************************************************
1616            ************************************************************************
1617            *
1618            *
1619 08D5E 06    KYPRCK RSTK=C               SAVE C(A)
1620 08D60 137          CD1EX
1621 08D63 D7           D=C    A            SAVE D1
1622            *
1623 08D65 113          A=R3                PTR TO FILE HEADER
1624 08D68 8E00         GOSUBL =GETPR
           00
1625            *
1626 08D6E 832          ?SB=0
1627 08D71 60           GOYES  KYPRC2       not SECURE?
1628 08D73 6000         GOTO   =MFERR       Err# loaded in C
1629            *
1630 08D77 DB    KYPRC2 C=D    A
1631 08D79 135          D1=C                RESTORE D1
1632 08D7C 07           C=RSTK              RESTORE C(A)
1633 08D7E 01           RTN
1634            *
1635            *
1636            ************************************************************************
1637            ************************************************************************
1638            **
1639            ** Name:(S) GTKYCD  -  Get Keycode
1640            ** Name:(S) GTKYC+  -  Get Keycode
1641            **
1642            ** Category:   EXCUTL
1643            **
1644            ** Purpose:  Evaluates string expression & returns keycode
1645            **
1646            **           The GTKYCD entry assumes that D0 points to the
1647            **           expression to be evaluated.  It errors if the
1648            **           string is null.
1649            **
1650            **           GTKYC+ assumes that the evaluated expression
1651            **           is already on the stack.  A status bit setting
```

```
1652                **          on entry indicates whether or not a null string
1653                **          should cause an error exit.
1654                **
1655                ** Entry:    2 entry points:
1656                **          1) GTKYCD - DO at expression.
1657                **          2) GTKYC+ - Evaluated string on stack.
1658                **                      S10=1 => Null string doesn't cause
1659                **                               error exit.
1660                **
1661                ** Exit:     CARRY CLR => B(A) = Keycode - between 1 & A8
1662                **                         A(A) = Shift value  (0,56,112)
1663                **
1664                **          If error encountered, error exits through MFERR
1665                **          with eDATTY or eIVARG
1666                **
1667                ** Calls:    EXPEXC, POP1S, DECHEX, CONVUC, DRANGE, MEMBER
1668                **
1669                ** Uses:
1670                **   Exclusive... A-D, D1,D0, S8,S9,S10
1671                **   Inclusive... Above + R0-R3, S0-S11, all of function scratch
1672                **
1673                ** Stack lvls: 5
1674                **
1675                ** History:
1676                **
1677                **    Date     Programmer   Modifications
1678                **   --------  ----------   -------------
1679                **  07/01/82   S.W.         Added documentation
1680                **  12/17/82   S.W.         When key assigned using ascii
1681                **                           char (not key#), now erroring on
1682                **                           alternate characters; for example
1683                **                           those with ascii val less than 32
1684                **                           (blank) or greater than 125 (}).
1685                **                           Was making assignments to keys in
1686                **                           non-obvious way.
1687                **  01/26/83   S.W.         Between ascii values 32 & 125 are
1688                **                           4 values which aren't represented
1689                **                           on our keyboard - these are now
1690                **                           trapped out.
1691                **
1692                **  02/22/83   B.S.         Changed GTKYC* entry point to
1693                **                           allow returning with carry set
1694                **                           and B(A)=0 if null string passed.
1695                **
1696                ************************************************************
1697                ************************************************************
1698                *
1699                *
1700 08D80 6B01     GTKYNM GOTO   GTKY50
1701                *
1702 08D84 135      GTKYC* D1=C
1703 08D87 1537           A=DAT1 W
1704 08D8B 85A            ST=1   10
1705 08D8E 6C00           GOTO   GTKYC+
1706                *
```

```
1707 08D92 8E00 =GTKYCD GOSUBL =EXPEXC
           00
1708 08D98 84A          ST=0   10          Null not legal
1709 08D9B 8E00 =GTKYC+ GOSUBL =POP1S      POP A STRING OFF STACK
           00
1710 08DA1 848          ST=0   8           SHIFT FLAG
1711 08DA4 849          ST=0   9           g SHIFT FLAG
1712 08DA7 D2           C=0    A
1713 08DA9 304          LCHEX  4
1714 08DAC 8BE          ?A>=C  A           MORE THAN 1 CHAR?
1715 08DAF 91           GOYES  GTKY05
1716              * 1 CHAR OR NULL STRING
1717 08DB1 CC           A=A-1  A
1718 08DB3 14F          C=DAT1 B           READ IN 1ST CHAR
1719 08DB6 D7           D=C    A
1720 08DB8 565          GONC   GTKY10      AT LEAST 1 CHAR?
1721              * NULL STRING
1722 08DBB D1           B=0    A           Keycode=0
1723 08DBD 87A          ?ST=1  10          Is null legal?
1724 08DC0 00           RTNYES             Yes, Return with carry set
1725 08DC2 8C00  GTINAR GOLONG =ARGERR
           00
1726              * MORE THAN 1 CHAR
1727 08DC8 06    GTKY05 RSTK=C             SAVE 00004
1728 08DCA D8           B=A    A           STRING LENGTH IN B(A)
1729 08DCC 1C1          D1=D1- 2
1730 08DCF 137          CD1EX
1731 08DD2 C2           C=C+A  A
1732 08DD4 135          D1=C               POINT TO 1ST CHARACTER
1733 08DD7 14B          A=DAT1 B           READ IT IN
1734 08DDA 3132         LCASC  \#\
1735 08DDE 962          ?A=C   B
1736 08DE1 F9           GOYES  GTKYNM
1737              *
1738 08DE3 07           C=RSTK             RESTORE 00004
1739 08DE5 8A5          ?B#C   A           MORE THAN 2 CHARS?
1740 08DE8 AD           GOYES  GTINAR
1741 08DEA 858          ST=1   8           2-CHAR STRING FLAG
1742 08DED 1C1          D1=D1- 2           PT TO 2ND CHARACTER
1743 08DF0 14F          C=DAT1 B           READ IN SHIFTED CHAR
1744 08DF3 D7           D=C    A           KEY TO BE ASSIGNED IN D(B)
1745 08DF5 8F00         GOSBVL =CONVUC
           000
1746              * 1ST CHAR BETTER BE F OR G
1747 08DFC 3164         LCASC  \F\
1748 08E00 962          ?A=C   B
1749 08E03 C0           GOYES  GTKY10
1750 08E05 859          ST=1   9           G SHIFT ON 2-CHAR STRING
1751 08E08 E6           C=C+1  A
1752 08E0A 966          ?A#C   B           NOT g SHIFT EITHER?
1753 08E0D 5B           GOYES  GTINAR
1754              *
1755 08E0F 3102  GTKY10 LCASC  \ \         Ascii Blank
1756 08E13 9E7          ?D<C   B
1757 08E16 CA           GOYES  GTINAR
```

```
1758 08E18 31D7          LCASC  \}\          Ascii }
1759 08E1C 9E3           ?D>C   B
1760 08E1F 3A            GOYES  GTINAR
1761 08E21 DB            C=D    A
1762 08E23 DA            A=C    A             Copy D(B) into A(B)
1763 08E25 37C5          LCHEX  7C605F5C      Disallow |\_ grave accent
           F506
           C7
1764 08E2F 27            P=     7
1765 08E31 8F00          GOSBVL =MEMBER
           000
1766 08E38 598           GONC   GTINAR
1767 08E3B 1F00          D1=(5) =KEYCOD       START OF KEYCODE MAP
           000
1768 08E42 D1            B=0    A
1769 08E44 D2            C=0    A
1770 08E46 171   GTKY15  D1=D1+ 2
1771 08E49 14F           C=DAT1 B
1772 08E4C E5            B=B+1  A             Keeps track of entry#
1773 08E4E 967           ?C#D   B             Not a match?
1774 08E51 5F            GOYES  GTKY15
1775               *
1776 08E53 3183          LC(2)  56
1777 08E57 868           ?ST=0  8             1-CHAR STRING?
1778 08E5A F1            GOYES  GTKY45
1779               * 2-CHAR STRING - MAP CHARACTER TO PRIMARY CASE
1780 08E5C 9ED   GTKY25  ?B<=C  B
1781 08E5F B0            GOYES  GTKY30
1782 08E61 E1            B=B-C  A
1783 08E63 58F           GONC   GTKY25        (B.E.T.)
1784               *
1785 08E66 6B5F  GTKY17  GOTO   GTINAR
1786               *
1787 08E6A 869   GTKY30  ?ST=0  9             F SHIFT?
1788 08E6D 40            GOYES  GTKY35
1789 08E6F C6    GTKY34  C=C+C  A
1790 08E71 C1    GTKY35  B=B+C  A             ACTUAL KEYCODE
1791 08E73 DA            A=C    A             0, 56, OR 112
1792 08E75 6180  GTKY40  GOTO   GTKY80
1793               *
1794               * D(A) MUST REFLECT PRIMARY/SHIFTED FOR 1-CHAR CASE, TOO
1795               *
1796 08E79 D0    GTKY45  A=0    A
1797 08E7B 9ED           ?B<=C  B             PRIMARY KEY?
1798 08E7E 7F            GOYES  GTKY40
1799               *
1800 08E80 E1            B=B-C  A
1801 08E82 9ED           ?B<=C  B             f SHIFTED KEY?
1802 08E85 CE            GOYES  GTKY35
1803 08E87 E1            B=B-C  A
1804 08E89 55E           GONC   GTKY34        (B.E.T.) - g SHIFTED KEY
1805               *
1806 08E8C 07    GTKY50  C=RSTK
1807 08E8E C6            C=C+C  A             MAX OF 4 CHARS (# <3 digits> )
1808 08E90 8B1           ?B>C   A
```

```
1809 08E93 3D          GOYES  GTKY17
1810            * ENSURE ALL OTHER CHARS ARE DIGITS
1811 08E95 81D         BSRB
1812 08E98 CD          B=B-1  A
1813 08E9A CD          B=B-1  A
1814 08E9C AF0         A=0    W
1815            *
1816 08E9F F0   GTKY55 ASL    A
1817 08EA1 1C1         D1=D1- 2
1818 08EA4 14B         A=DAT1 B
1819 08EA7 8F00        GOSBVL =DRANGE
          000
1820 08EAE 47B  GTKY60 GOC    GTKY17        NOT A DIGIT?
1821 08EB1 BE0         ASL    B
1822 08EB4 A0D         B=B-1  P              DECREMENT COUNTER
1823 08EB7 57E         GONC   GTKY55
1824 08EBA F4          ASR    A
1825 08EBC 8F00        GOSBVL =DECHEX        CONVERT TO HEXADECIMAL
          000
1826 08EC3 8A8         ?A=0   A
1827 08EC6 0A          GOYES  GTKY17
1828            *
1829 08EC8 D2          C=0    A
1830 08ECA D1          B=0    A
1831 08ECC DC          ABEX   A              B(A) contains keycode; A(A)=0
1832 08ECE 318A        LC(2)  56*3
1833 08ED2 8B1         ?B>C   A              Invalid keynumber?
1834 08ED5 19          GOYES  GTKY17         No => eIVARG
1835 08ED7 3183        LC(2)  56             Mod 56 reduction removes f/g shift
1836 08EDB 8B8  GTKY65 ?B<=A  A              Can B be reduced mod 56
1837 08EDE 70          GOYES  GTKY70         No, then okay
1838 08EE0 CA          A=A+C  A              Add it to A
1839 08EE2 58F         GONC   GTKY65         (B.E.T.) Loop back
1840            *-
1841 08EE5 EA   GTKY70 A=A-C  A
1842 08EE7 304         LC(1)  52             BOTTOM OF EOL KEY (ROW MAJOR)
1843 08EEA C2          C=C+A  A              ADD SHIFT
1844 08EEC 965         ?B#C   B              NOT the bottom of EOL key?
1845 08EEF 80          GOYES  GTKY80         Yes=>    convert to top of EOL key
1846            *
1847 08EF1 31E0        LC(2)  14             Adjustment from bottom to top
1848 08EF5 E1          B=B-C  A              Do adjustment
1849 08EF7 31C2 GTKY80 LC(2)  44             Return with carry clear
1850 08EFB C2          C=C+A  A
1851 08EFD 961         ?B=C   B              f SHIFT?
1852 08F00 EA          GOYES  GTKY60
1853 08F02 E6          C=C+1  A
1854 08F04 961         ?B=C   B              g SHIFT?
1855 08F07 7A          GOYES  GTKY60
1856 08F09 03          RTNCC
```

```
1857                    STITLE PSHGSB,PSHUPD,PSHMCR
1858            ****************************************************************
1859            ****************************************************************
1860            **
1861            ** Name:(S) PSHGSB  -  Push address on GOSUB Stk
1862            ** Name:(S) PSHUPD  -  Push address on GOSUB Stk
1863            ** Name:(S) PSHMCR  -  Push address on GOSUB Stk
1864            **
1865            ** Category:   GENUTL
1866            **
1867            ** Purpose:
1868            **      Push address and return type nibble on GOSUB stack
1869            **      Allows address to be updated when memory moves
1870            **      Allows microcode GOSUB/RETURN to work
1871            **
1872            ** Entry:
1873            **      A(A)   =  Address to push on stack
1874            **      PSHMCR:   Sets return type for Microcode return
1875            **      PSHUPD:   Sets up as Update address. P must be 0.
1876            **      PSHGSB:   C(S) = Return type (see GOSUB)
1877            **
1878            ** Exit:
1879            **      Carry Clear:
1880            **        P      =  0     (not necessary for PSHGSB)
1881            **        D1 @ Return type nibble on stack
1882            **        C[0]  = Return type
1883            **        C[5-1]=Address just pushed on stack
1884            **      Error Exit:
1885            **        Insufficient Memory to open stack
1886            **
1887            ** Calls:      PSHSTK
1888            **
1889            ** Uses.......
1890            **  Exclusive: C(W),D(S),P,D
1891            **  Inclusive: A-D,D0,D1
1892            **
1893            ** Stk lvls:   3
1894            **
1895            ** Algorithm:
1896            **
1897            ** PSHMCR:     C(S) <-- Microcode Return type
1898            ** PSHUPD:     C(S) <-- Update Address Return type
1899            ** PSHGSB:     Save Return Type  D(S) <-- C(S)
1900            **             Save Return address on stack
1901            **             Open up GOSUB stack by 6 nibbles   (PSHSTK)
1902            **             Restore address and return type
1903            **             Write return type and address to stack
1904            **             RTNCC
1905            **
1906            ** History:
1907            **
1908            **    Date     Programmer             Modification
1909            **  --------   ----------   ----------------------------------
1910            **  09/30/82   J.P.         Added code
1911            **
```

```
1912            ****************************************************************
1913            ****************************************************************
1914 08F0B 29  =PSHMCR P=     9              Microcode Return type
1915            *
1916 08F0D 0D  =PSHUPD P=P-1                 Update address Return type
1917 08F0F 80FF        CPEX   15             C(S) <-- Return type
1918 08F13 20  =PSHGSB P=     0
1919 08F15 AC7         D=C    S              Save Return type
1920 08F18 D6          C=A    A
1921 08F1A 06          RSTK=C                Save Return address
1922 08F1C D2          C=0    A
1923 08F1E 306         LCHEX  6              # nibbles to open up
1924 08F21 D5          B=C    A
1925 08F23 22          P=     2              Adj. 3 ptrs:GSBSTK,FORSTK,MTHSTK
1926 08F25 1B3A        DO=(5) =GSBSTK        Top of GOSUB stack
           5F2
1927 08F2C 7F4D        GOSUB  PSHSTK
1928 08F30 07          C=RSTK                Restore Return address
1929 08F32 ACB         C=D    S              Restore Return type
1930 08F35 812         CSLC                  Move to C(0)
1931 08F38 15D5        DAT1=C 6              Write Return type & addr to stack
1932 08F3C 03          RTNCC
1933            *
```

```
1934                    STITLE POPSTK
1935            ********************************************************************
1936            ********************************************************************
1937            **
1938            ** Name:(S) POPSTK  -  Pop Stack
1939            ** Name:    POPGSB  -  Pop Stack
1940            ** Name:(S) POPUPD  -  Pop Stack
1941            **
1942            ** Category:   GENUTL
1943            **
1944            ** Purpose:  Deletes stack entry(ies) and adjusts pointers
1945            **         --pertains to FOR/NEXT, GOSUB, etc.
1946            **           POPGSB/POPUPD:
1947            **           Pop return address/update address off GOSUB stack
1948            **         --Reads Return Address and Return type, then deletes
1949            **
1950            ** Entry:
1951            **  POPGSB:  Sets C(A) and A(A) to top entry of GOSUB stack
1952            **           Reads Return type and Return address into D
1953            **           Sets P for PTRADJ
1954            **  POPSTK:  C(A) points to start of entry to delete (pop)
1955            **           A(A) points to end of entry to delete
1956            **           P set for PTRADJ
1957            **
1958            ** Exit:
1959            **  POPSTK: CARRY CLEAR, P=0.
1960            **  POPGSB/POPUPD: If Carry set
1961            **                    Stack was empty, P unchanged
1962            **                 Else carry clear, P=0
1963            **                   D(A) = Return address
1964            **                   D(S) = Return type (see RETURN)
1965            **
1966            **                   If the address on the stack points into a
1967            **                   file and that file is purged before the
1968            **                   address is popped off, the return address
1969            **                   will be ZERO.
1970            **
1971            **                   This can happen if Expression Execute is
1972            **                   called, and a multi-line user defined
1973            **                   issues a PURGE.
1974            **
1975            **                   Calling routines may need to check for this.
1976            **
1977            **           via PTRAD1
1978            **
1979            ** Calls:    MOVED3, RTNSTK
1980            **
1981            ** Uses:     A, B(0-5), C, D1, D0
1982            **           If POPGSB/POPUPD uses D(W)
1983            **           B(S) must be preserved for POLL
1984            **
1985            ** Detail:   Could also be useful to variable elimination
1986            **           (e.g. DESTROY) or to eliminating SUB environments
1987            **           Must immediately precede PTRAD1
1988            **
```

```
1989            **          If the return address on the stack points into a
1990            **          file and that file is purged before the address is
1991            **          is popped off, this address will be ZERO.
1992            **
1993            **          This can happen if Expression Execute is called and
1994            **          a user defined function issues a PURGE.
1995            **          A calling routine may have to check this is EXPEXC
1996            **          can be called in the interim.
1997            **
1998            ** Stack Lvls: 1
1999            **
2000            ** History:
2001            **
2002            **    Date     Programmer   Modifications
2003            **  --------   ----------   -------------
2004            **  07/04/82   S.W.         Added documentation
2005            **  08/10/82   S.W.         Modified to preserve math stk
2006            **  10/06/82   J.P.         Added POPGSB/POPUPD entries
2007            **  10/07/82   NM           Added stack-empty check
2008            **  02/10/83   J.P.         Use only B(0-5) to pres B(S)
2009            **
2010            ****************************************************************
2011            ****************************************************************
2012            *
2013            *
2014 08F3E      =POPGSB
2015 08F3E 7C50 =POPUPD GOSUB  RTNSTK
2016 08F42 400          RTNC
2017 08F45 AF7          D=C    W              Save them
2018 08F48 817          DSRC                  D(S) <- Rtn type, D(A) <- Rtn Addr
2019 08F4B D6           C=A    A              Set C = Start of entry to delete
2020 08F4D 165          DO=DO+ 6              Position to end of entry
2021 08F50 132          ADOEX                 Set A = End of entry to delete
2022 08F53 22           P=     2              Adjust 3 pointers
2023            *
2024            * Do Not Destroy B(S) for POLL
2025            *
2026 08F55 1B99 =POPSTK DO=(5) =AVMEME
           5F2
2027 08F5C 80F5         CPEX   5
2028 08F60 25           P=     5
2029 08F62 A95          B=C    WP             SAVE END SOURCE & P
2030 08F65 E8           B=B-A  A              -OFFSET
2031 08F67 F9           B=-B   A
2032 08F69 131          D1=A                  END DESTINATION
2033 08F6C 142          A=DAT0 A              BEGIN SOURCE
2034 08F6F 134          DO=C                  END SOURCE
2035 08F72 E2           C=C-A  A              SOURCE LENGTH
2036 08F74 8E00         GOSUBL =MOved3        DELETE
           00
2037 08F7A 25           P=     5              B(5) = original P setting
2038 08F7C A89          C=B    P
2039 08F7F 80F5         CPEX   5              Restore # ptrs
2040            * Falls directly into PTRAD1
2041            ****************************************************************
```

```
2042            ****************************************************************
2043            **
2044            ** Name:    PTRAD1  -  Pointer Adjust
2045            ** Name:    PTRAD2  -  Pointer Adjust
2046            **
2047            ** Category:   PTRUTL
2048            **
2049            ** Purpose:
2050            **      Adds specified offset to specified number of sequential
2051            **      RAM pointers
2052            **
2053            ** Entry:
2054            **      P=   (n-1) the number of pointers to adjust,
2055            **             e.g. if P=0, only 1 pointer will be adjusted
2056            **      B(A) offset to add to each pointer
2057            **      2 entry points:
2058            **             1) PTRAD1 -  AVMEME is the 1st pointer in the
2059            **                          block of sequential pointers to be
2060            **                          adjusted.
2061            **             2) PTRAD2 -  Assumes 1st pointer to be adjusted
2062            **                          is pointed to by D0
2063            **
2064            ** Exit:
2065            **      P      =  0
2066            **      Carry clear
2067            **      D0 5 nibbles past last pointer adjusted
2068            **      C(A)   =  adjusted value of last pointer updated
2069            **      B(A) is preserved
2070            **
2071            ** Calls:      none
2072            **
2073            ** Uses.......
2074            **  Inclusive: C(A), D0, P
2075            **
2076            ** Stk lvls:   0
2077            **
2078            **
2079            ** History:
2080            **
2081            **    Date     Programmer             Modification
2082            **  --------   ----------   ----------------------------------
2083            **  07/04/82   S.W.         Added documentation
2084            **
2085            ****************************************************************
2086            ****************************************************************
2087            *
2088 08F83 1B99 =PTRAD1 D0=(5) =AVMEME
           5F2
2089 08F8A 146  =PTRAD2 C=DAT0 A              CURRENT PTR VALUE
2090 08F8D C9           C=C+B  A              INCREMENT (OR DECREMENT)
2091 08F8F 144          DAT0=C A              & REPLACE
2092 08F92 164          D0=D0+ 5              POINT TO NEXT PTR
2093 08F95 0D           P=P-1
2094 08F97 52F          GONC   PTRAD2
2095 08F9A 20           P=     0
```

```
2096 08F9C 03          RTNCC
2097           *
2098 08F9E 1B3A  RTNSTK DO=(5) =GSBSTK
          5F2
2099 08FA5 142         A=DAT0 A
2100 08FA8 164         DO=DO+ 5
2101 08FAB 146         C=DAT0 A
2102 08FAE 8BE         ?A>=C  A              Empty stack?
2103 08FB1 00          RTNYES
2104           *
2105 08FB3 130         DO=A
2106 08FB6 15E5        C=DAT0 6
2107 08FBA 03          RTNCC
2108           *
2109 08FBC 3210  stmbfd LC(3)  =bSTMT
          8
2110 08FC1 6000        GOTO   =i/ofnd
2111           *
```

```
2112                    STITLE RETURN, POP
2113            ********************************************************************
2114            ********************************************************************
2115            **
2116            ** Name:     RETURN  -  Statement Execute
2117            ** Name:     POP     -  Statement Execute
2118            **
2119            ** Category:   STEXEC
2120            **
2121            ** Purpose:
2122            **      Execute RETURN or POP statement
2123            **
2124            ** Entry:
2125            **      P       =  0
2126            **      D0 past RETURN/POP token
2127            **
2128            ** Exit:
2129            **      P       =  0
2130            **      If BASIC RETURN
2131            **         D0 @ Address popped off top of GOSUB stack
2132            **         Return to Run Loop
2133            **      If BASIC RETURN to Keyboard
2134            **         Update Current Line @ RETURN
2135            **         Clear Program Annunc and flag
2136            **         If Statement buffer NOT empty
2137            **            D0 @ Address popped off
2138            **            Return to Run Loop
2139            **         else
2140            **            Return to Main Loop through BASIC Loop exit
2141            **      If POP
2142            **         Address popped off stack
2143            **         Return through NXTSTM
2144            **      If Microcode Return
2145            **         If RETURN
2146            **            Return to Microcode through Hardware stack
2147            **         If POP
2148            **            No-op
2149            **      If Boundary Address
2150            **         If RETURN
2151            **            Error ---> Return w/o GOSUB
2152            **         If POP
2153            **            No-op
2154            **
2155            **      Error/Exceptions:
2156            **         If RETURN
2157            **            If Empty stack
2158            **               eRwoGS
2159            **            If Special Return Type and No POLL response
2160            **               eMMCOR
2161            **            If Return type #0-4
2162            **               eMMCOR
2163            **         If POP
2164            **            No-ops, never errors
2165            **
2166            ** Calls:      FPOLL, POPSTK, SCOPCK, RACTMI, KBRTCK, TRTOr
```

```
2167            **            UPDPCA, RTNSTK, STMBUF, SNcrlf, TRFLCK, TRCFRM
2168            **
2169            ** Uses.......
2170            **  Exclusive: A,B,C,D,D0,D1,sRETRN (S0),R2
2171            **  Inclusive: A-D,D0,D1,sRETRN,R1,R2,R3,S13
2172            **
2173            **  R2 = Saved Return type and address
2174            **
2175            ** Stk lvls:   5
2176            **
2177            ** NOTE:
2178            **      RETURN from Keyboard must collapse the Stmt Buffer
2179            **        A zero length Statement Buffer prevents returning to
2180            **        the keyboard into a different Statement Buffer than
2181            **        the buffer originally issuing the GOSUB
2182            **      CONT must also do this
2183            **
2184            **      Scenario:
2185            **        GOSUB issued from keyboard
2186            **        Return address points back into Statement Buffer
2187            **        Subroutine is PAUSED
2188            **        Statements entered from keyboard destroy old buffer
2189            **        Any statement that can CONTINUE the subroutine and
2190            **          cause it to return, must empty the Stmt Buffer:
2191            **          RETURN, CONT, CONT Key, SST
2192            **
2193            ** Detail:
2194            **      Return Types:
2195            **
2196            **      0       Return to Program
2197            **      1       Return to Keyboard
2198            **      2       ON TIMER#1 GOSUB
2199            **      3       ON TIMER#2 GOSUB
2200            **      4       ON TIMER#3 GOSUB
2201            **      8       Microcode Return
2202            **      9-E     Special Return types: future stmt extensions
2203            **      F       Boundary Address
2204            **                If Address=0
2205            **                   Environment Boundary
2206            **                else
2207            **                   Update Address
2208            **
2209            **      Stack order:  Low --> High Memory
2210            **            Return Type     1 nibble
2211            **            Return Address  5 nibbles
2212            **
2213            ** Algorithm:
2214            **
2215            ** POP: Clear Return flag                          (sRETRN)
2216            **      goto 1;
2217            ** RTN: Set Return flag                            (sRETRN)
2218            **      If RETURN from Keyboard                 (not PgmRun)
2219            **         Collapse Statement Buffer                (I/OCOL)
2220            **   1: If empty stack
2221            **   2:    If POP
```

```
2222            **            Return through Next Statement        (NXTSTM)
2223            **         else
2224            **            Error --> RTN w/o GOSUB
2225            **       Set D0 @ Top of GOSUB Stack
2226            **       Read Return type and Return address
2227            **       Shift Return type to A(S)
2228            **       Save Return type and address               (R2)
2229            **       If Boundary address                        (A(S)=F)
2230            **          Dont' POP addresss                      (goto 2)
2231            **       If Special return type                     (A(S)>=8)
2232            **          If Microcode return address             (A(S)=8)
2233            **             If POP
2234            **                Return to Next statement
2235            **          else
2236            **             POLL on Return Type                  (POLL)
2237            **       If ON ERROR return   (Rtn addr = ERRSUB)
2238            **          Clear ERRSUB address
2239            **       Adjust pointers & collapse GOSUB stack     (POPGSB)
2240            **       If POP
2241            **          goto Next Statement
2242            **       Restore Return address and return type
2243            **       If Special Return type                     (A(S)>=8)
2244            **          If Microcode Return                     (A(S)=8)
2245            **             RSTK <-- Return address
2246            **             RTN
2247            **          else
2248            **             Error Exit --> System Error         (CORUPT)
2249            **       If Return to Program                       (A(S)=0)
2250            **          If within current program scope         (SCOPCK)
2251            ** 3:          Set PgmRun
2252            **             Restore Return address              (R2)
2253            **             Set D0 @ Return address
2254            ** 4:          Return to Run loop                  (NXTST5)
2255            **          else
2256            **             Error --> System Error              (CORUPT)
2257            **       If ON TIMER GOSUB return                   (A(S)=2,3,4)
2258            **          Reactivate Timer Interval               (RACTMI)
2259            **          go Restore Rtn address and Run          (goto 3)
2260            **       If Return to Keyboard                      (A(S)=1)
2261            **          If Statement buffer empty               (KBRTCK)
2262            **             Exit BASIC loop                      (EXITRN)
2263            **          else
2264            **             Return to keyboard (goto 4)         (NXTST5)
2265            **
2266            ** History:
2267            **
2268            ** Date      Programmer        Modification
2269            ** --------  ----------        ---------------------------------
2270            ** 03/07/83  J.P               Deleted NOPRGM call, see KBRTCK
2271            ** 03/18/83  J.P.              Call SFGPGM if return to Program
2272            ** 04/25/83  J.P.              CLRST when exiting: EXITRN
2273            ** 05/17/83  J.P.              Call TRTOr @ RTNX70
2274            ** 05/19/83  J.P.              Set PCADDR @ Rtnaddr 'cus RETURN
2275            **                            from keyboard into Prgm now SUSPs
2276            ** 05/19/83  J.P.              Remove SFGPGM call; since RETURN
```

```
2277            **                              from keyboard will SUSP now, but
2278            **                              must set PgmRun so SUSP happens
2279            ** 05/27/83  J.P.               Return to keyboard pop & restores
2280            **                              CNTADR off GOSUB stack
2281            ** 09/12/83  J.P.               Set PCADDR @ Line length of Rtn
2282            **                              address:  SR#1040-5
2283            **
2284            ****************************************************************
2285            ****************************************************************
2286            ****************************************************************
2287            ****************************************************************
2288            **
2289            ** Name:(S) pRTNTp  -  Poll on Special Return type
2290            **
2291            ** Category:   POLL
2292            **
2293            ** Type:       FPOLL
2294            **
2295            ** Purpose:
2296            **      Poll for Special Return type
2297            **      Allow for future extension of Special Return types on
2298            **        the GOSUB stack.  When the RETURN is encountered,
2299            **        a LEX file may handled to do something before the
2300            **        RETURN (ex: Reactivate a Timer)
2301            **
2302            **      Return types: 9-E are reserved for future implementation
2303            **      The GOTO+ entry point allows the special Return type to
2304            **        be passed on entry in R3(S)
2305            **
2306            ** Should poll be "Handled" (return with XM=0)?:
2307            **      No - if this poll is handled, it is a take over pollxxx
2308            **
2309            ** Meaning of "Handling" Poll (what does code do if handled?):
2310            **      Do the appropriate "special" return processing
2311            **      Pop address of GOSUB stack
2312            **      Perform the RETURN or POP
2313            **
2314            ** Entry conditions for handler (registers, ST, RAM, etc.):
2315            **      --Carry set on entry iff fastpoll--
2316            **      B[A] = Poll number (pRTNTp)
2317            **      HEX mode.
2318            **      P=0.
2319            **      R2(S) = Return type  (Range = 9-E)
2320            **      R2(A) = Return address
2321            **      sRETRN  (S0) = 1 if RETURN
2322            **                   = 0 if POP
2323            **      The address is NOT popped off the stack
2324            **
2325            **      DO NOT destroy S0 or R2 while determining if handling
2326            **
2327            ** Normal exit conditions from handler if handled (ST, RAM,
2328            ** registers, etc.):
2329            **      HEX mode.
2330            **      Perform the "special" processing
2331            **      Pop the address off stack  (GOSBVL =POPGSB)
```

```
2332          **      If POP
2333          **         GOVLNG NXTSTM
2334          **      If RETURN
2335          **         If Return to Program (type must indicate this)
2336          **            Set PgmRun
2337          **            Save return address in R2
2338          **            Set D0 @ return address
2339          **            If TRACE needed                    (TRFLCK)
2340          **               TRACE TO                         (TRTO+)
2341          **            Set D0 @ Return address               (R2)
2342          **            go execute "Return stmt"    (govlng RUNRT1)
2343          **         If Return to Keyboard
2344          **            If tracing                         (TRFLCK)
2345          **               Send CR/LF                      (CRLFSD)
2346          **            If Keyboard buffer to return to    (KBRTCK)
2347          **               Set D0 @ Return address         (R2)
2348          **               go execute "Return stmt" (govlng RUNRT1)
2349          **
2350          **
2351          **      Sample code:
2352          **
2353          **            GOSBVL   =POPGSB      Pop addr off stack
2354          **            C=D      A
2355          **            R2=A                  Save Return addres
2356          **            ?ST=0    sRETRN       POP?
2357          **            GOYES    RTN40
2358          **            ?ST=1    sRTNKY       Return to Keyboard ?
2359          **            GOYES    RTN20
2360          **            ST=1     PgmRun       Set Pgm Running flag
2361          **            A=R2                  Return address
2362          **            D0=A
2363          **            GOSBVL   =TRFLCK      Tracing ?
2364          **            GOC      RTN10        No
2365          **            GOSBVL   =TRTO+       TRACE TO
2366          **   RTN10    A=R2
2367          **            D0=A                  D0 @ Return address
2368          **            GOVLNG   =RUNRT1      Execute Return stmt
2369          **
2370          **   * Return to keyboard
2371          **
2372          **   RTN20    GOSBVL   =TRFLCK      Tracing ?
2373          **            GOC      RTN30
2374          **            GOSBVL   =CRLFSD      Send CR/LF
2375          **   RTN30    GOSBVL   =KBRTCK      Keyboard buffer?
2376          **            GOTO     RTN10        Yes, go execute
2377          **   *
2378          **   * POP
2379          **   *
2380          **   RTN40    GOVLNG   =NXTSTM
2381          **
2382          **
2383          ** Normal exit conditions from handler if not handled (ST, RAM,
2384          ** registers, etc.):
2385          **      HEX mode.
2386          **      XM=1.
```

```
2387            **      sRETRN (S0) and R2 must be be preserved--
2388            **
2389            ** Available subroutine levels: 7
2390            **      --FPOLL handler is two levels deeper than caller--
2391            **      RETURN/POP is statement execute: all levels available
2392            **
2393            ** NOTE:
2394            **      See GOTO+ entry for pushing special return type on
2395            **          GOSUB stack
2396            **      The return type must NOT conflict with other
2397            **          GOSUB/RETURN extended statements
2398            **      The return type or somewhere else --- must reflect
2399            **          if return to PROGRAM or KEYBOARD. This is
2400            **          determined at GOSUB time from PgmRun flag
2401            **
2402            ** What registers/RAM may be used if handled?:
2403            **      --A-D, D0, D1, P always available--
2404            **      --Statement execute usage
2405            **
2406            ** What registers/RAM may be used if not handled?:
2407            **      --A-C, D[15-5] D0, D1, P always available (FPOLL only)--
2408            **      --NOTE: D[A] is sacred in FPOLL!!--
2409            **      --R0,R1,R3,R4
2410            **
2411            ** Envisioned application(s):
2412            **      Special GOSUB statement:
2413            **        ON INTERUPT GOSUB....
2414            **        ON ALARM GOSUB....
2415            **        ON ... GOSUB....
2416            **
2417            **        where "something" must be done before the actual
2418            **        return is exeucte.  For example, schedule an ALARM
2419            **
2420            ** History:
2421            **
2422            **    Date     Programmer             Modification
2423            **  --------   ----------   ---------------------------------
2424            **  05/02/83   J.P.         Changed to Fast Poll
2425            **
2426            ****************************************************************
2427            ****************************************************************
2428            *
2429            ****************************************************************
2430            ****************************************************************
2431            **
2432            ** Name:(S) CORUPT  -  Report "System Error" error
2433            **
2434            ** Category:   SYSTEM
2435            **
2436            ** Purpose:
2437            **      To report "System Error" as an execution error.
2438            **
2439            ** Entry:
2440            **      P      =  0
2441            **      S13=0 if not a running program (i.e., keyboard
```

```
2442            **        execution error)
2443            **      S13=1 if running program
2444            **      No other necessary conditions.
2445            **
2446            ** Exit:
2447            **      Exits to BASIC main loop (ERRRTN)
2448            **
2449            ** Calls:      MFERR
2450            **
2451            ** Uses....... BASIC main loop can use anything
2452            **
2453            ** Stk lvls:   BASIC main loop can use anything
2454            **
2455            ** NOTE:
2456            **      Setting P=0 selects the following error options:
2457            **        -- not a parse error
2458            **        -- store ERRN (and ERRL if S13=1)
2459            **        -- display "ERR:" (or "ERR L<#>:")
2460            **
2461            ** Detail:
2462            **     =CORUPT LC(2)  =eMMCOR
2463            **             GOLONG =MFERR
2464            **
2465            ** History:
2466            **
2467            **    Date     Programmer              Modification
2468            **  --------   ----------   ----------------------------------
2469            **  11/09/83   MB           Documentation
2470            **
2471            ********************************************************************
2472            ********************************************************************
2473            *
2474            * POP statement entry
2475            *
2476 08FC5 0000         REL(5) =STOPDC
           0
2477 08FCA 0000         REL(5) =RTNCC
           0
2478 08FCF 840  =POP    ST=0   sRETRN         Clear Return flag
2479 08FD2 6F10         GOTO   RTNX10
2480            *
2481            * RETURN statement entry
2482            *   If RETURN from Keyboard
2483            *      Collapse Statement Buffer
2484            *
2485 08FD6 0000         REL(5) =STOPDC
           0
2486 08FDB 0000         REL(5) =RETRNp
           0
2487 08FE0 850  =RETURN ST=1   sRETRN         Set Return flag
2488 08FE3 78F0         GOSUB  STMBUF         Collapse stmt buffer if executed
2489 08FE7 718C         GOSUB  trflck
2490 08FEB 460          GOC    RTNX10
2491 08FEE 766C         GOSUB  trcfrm
2492            *                               from keyboard
```

```
2493            * Check for entpy GOSUB stack
2494            *
2495 08FF2 78AF  RTNX10 GOSUB  RTNSTK
2496 08FF6 532          GONC   RTNX30          Non-empty?
2497            *
2498            * If POP
2499            *    Return through NXTSTM  (noop)
2500            * else
2501            *    Error ---> RTN w/o GOSUB
2502            *
2503
2504 08FF9 860   RTNERR ?ST=0  sRETRN          POP statement ?
2505 08FFC E0           GOYES  nxtstm
2506 08FFE 707C         GOSUB  SNcrlf          SEND CR/LF IF IN TRACE FLOW MODE
2507 09002 3100         LC(2)  =eRwoGS         RTN w/o GOSUB
2508 09006 6000         GOTO   =MFERR
2509 0900A 6D3A  nxtstm GOTO   NXTSTM          Return - No op
2510            *
2511            * Special Return type --- Unrecognizable
2512            *
2513 0900E 8E00  RTNX20 GOSUBL =FPoll
           00
2514 09014 A3           CON(2) =pRTNTp         POLL on Return type
2515 09016 62EF         GOTO   RTNERR          Error if no response
2516            *
2517            * Set D0 @ Top of GOSUB stack
2518            * Read Return type and Return address
2519            * Shift Return type and save both
2520            * Check for Boundary address
2521            *
2522 0901A AFA   RTNX30 A=C    W               Read in RETURN type & address
2523 0901D 814          ASRC                   Move Return type to A(S)
2524 09020 102          R2=A                   Save Return type and address
2525 09023 B44          A=A+1  S               Boundary address ?
2526 09026 42D          GOC    RTNERR          Don't pop address
2527            *
2528            * Check for Special Return type
2529            *
2530 09029 A4C          A=A-1  S               Restore Return type
2531 0902C A44          A=A+A  S               Special Return type >= 8 ?
2532 0902F 5C0          GONC   RTNX40          No
2533 09032 94C          ?A#0   S               Not Microcode Return ?
2534 09035 9D           GOYES  RTNX20          go POLL on Return type
2535            *
2536            * If Microcode Return and POP statement
2537            *    Return ... No-op
2538            *
2539 09037 860          ?ST=0  sRETRN          POP ?
2540 0903A 0D           GOYES  nxtstm
2541
2542            *
2543            * Check if Return Address from ON ERROR-GOSUB execution
2544            *   Read Return address for ON ERROR=GOSUB statement
2545            *   If Return address = ON ERROR-GOSUB Return address
2546            *      Zero ON ERROR-GOSUB Return address   (ERRSUB)
```

```
2547                *
2548 0903C 1F38     RTNX40 D1=(5) =ERRSUB
           6F2
2549 09043 147              C=DAT1 A            Read ON ERROR-GOSUB Return address
2550 09046 8A6              ?C#A   A            Return addresses NOT equal ?
2551 09049 70               GOYES  RTNX45
2552 0904B D2               C=0    A            Zero ON ERROR-GOSUB Return address
2553 0904D 145              DAT1=C A
2554                *
2555                * Collapse GOSUB stack
2556                * If POP
2557                *    golong NXTSTM
2558                *
2559 09050 7AEE     RTNX45 GOSUB  POPGSB        COLLAPSE STK ENTRY, ADJ PTRS
2560 09054 860              ?ST=0  sRETRN       POP statement ?
2561 09057 3B               GOYES  nxtstm
2562                *
2563                * Restore Return type and return address
2564                * Check for Microcode Return
2565                * If Special Return type and NOT microcode return
2566                *    Error --> Corrupt
2567                *
2568 09059 112              A=R2                Restore return  type and addr
2569 0905C AF6              C=A    W
2570 0905F A46              C=C+C  S
2571 09062 5F0              GONC   RTNX50       Not special return type
2572 09065 94E              ?C#0   S            Not microcode return ?
2573 09068 B1               GOYES  CORUPT       Error exit
2574 0906A 740C             GOSUB  SNcrlf       Send CR/LF if in Trace Flow
2575 0906E 06               RSTK=C
2576 09070 01               RTN                 Return to microcode
2577                *
2578                * If Return to Program
2579                *    If not with current program scope
2580                *       Error --> System Error
2581                *
2582 09072 A4C      RTNX50 A=A-1   S            Return to Program = 0
2583 09075 551              GONC   RTNX60
2584 09078 7FD0             GOSUB  SCOPCK       Within current program scope ?
2585 0907C 542              GONC   RTNX70       Yes
2586 0907F 7FEB             GOSUB  SNcrlf
2587 09083 3100 =CORUPT     LC(2)  =eMMCOR
2588 09087 6000             GOTO   =MFERR       No -- Error Exit
2589                *
2590                * If ON TIMER GOSUB return
2591                *    A(S) = Timer#
2592                *    Reactive Timer interval
2593                * else
2594                *    Error --> Corrupt
2595                *
2596 0908B 948      RTNX60 ?A=0    S            Return to Keyboard ?
2597 0908E 33               GOYES  RTNX80
2598 09090 304              LCHEX  4
2599 09093 816              CSRC
2600 09096 9C6              ?A>C   S            Not ON TIMER ??
```

```
2601 09099 AE          GOYES  CORUPT
2602 0909B 8E00        GOSUBL =RACTMI        Reactivate timer interval
           00
2603               *
2604               * Return to running program
2605               *   Restore PCADDR @ return address incase
2606               *     RETURN from keyboard after "SUSP" program
2607               *     will stop execution after the RETURN
2608               *   PCADDR will point @ statement returned to
2609               *   FETCH will show this line NOT the "GOSUB" invoking line
2610               *
2611               *   Fix to SR#1040-5
2612               *     Set PCADRR @ LINE LENGTH of Return Address
2613               *
2614               *   Set PgmRun flag so SUSP will occur
2615               *   Must do this AFTER "trflck" so TRACE won't happen
2616               *
2617 090A1 11A   RTNX70 C=R2
2618 090A4 8E00        GOSUBL =UPDPCA        Update PCADDR @ line length
           00
2619 090AA 134         DO=C
2620 090AD 7BBB        GOSUB  trflck
2621 090B1 480         GOC    RTNX75
2622 090B4 8E00        GOSUBL =TRTOr         TRACE
           00
2623 090BA 85D   RTNX75 ST=1   PgmRun        Set Program Running flag
2624 090BD 6B10        GOTO   RTNX90         Return to Run Loop
2625               *
2626               * Return to Keyboard
2627               *   Pop CNTADR off GOSUB stack and restore it
2628               *   If Statement buffer empty
2629               *      Don't RETURN, goto BSCEXT
2630               *   else
2631               *      Restore Return address
2632               *
2633 090C1 7DAB  RTNX80 GOSUB  SNcrlf        Send CR/LF to display
2634 090C5 757E        GOSUB  POPUPD         Pop CNTADR off GOSUB stack
2635 090C9 DB          C=D    A
2636 090CB 1BE7        DO=(5) =CNTADR
           6F2
2637 090D2 144         DATO=C A              Restore CNTADR
2638 090D5 7A30        GOSUB  KBRTCK
2639 090D9 8C50  RTNX90 GOLONG NEXT50
           8F
2640               *
2641               *
2642               ********************************************************************
2643               ********************************************************************
2644               **
2645               ** Name:(S) STMBUF  -  Collapse statement buffer check
2646               ** Name:(S) STMBCL  -  Collapse statement buffer check
2647               **
2648               ** Category: EXCUTL
2649               **
2650               ** Purpose : Some statements need to collapse the statement
```

```
2651            **          buffer when executed from the keyboard.
2652            **          These statements are: CONT, RETURN, ENDSUB, ENDDEF
2653            **           They call the entry point STMBUF.
2654            **
2655            **          STMBUF - Collapses Statement Buffer only if no
2656            **                   program is running
2657            **          STMBCL - Collapses Statement Buffer, unconditionally
2658            **
2659            ** Entry : S13 = 0 if the statement is executed from keyboard
2660            **         STMBCL: Alway collapses
2661            **
2662            ** Exit : Carry set
2663            **
2664            ** Calls : I/OCOL, STMBFD        May exit via FORUPD
2665            **
2666            ** Uses : A-D, D0, D1, S14 (STMBUF entry only)
2667            **
2668            ** Stk lvls : 2
2669            **
2670            ** History:
2671            **
2672            **    Date     Programmer   Modifications
2673            **  --------   ----------   -------------
2674            **  01/27/83   S.W.         Added call to RFADJ- to zero
2675            **                           references to collapsed buffers.
2676            **                           Additionally uses R0,R1
2677            **
2678            **  05/19/83   J.P.         Set NoCont if not running so
2679            **                           ENDSUB,ENDDEF,RETURN will SUSP
2680            **
2681            ********************************************************************
2682            ********************************************************************
2683            *
2684 090DF 87D  =STMBUF ?ST=1  PgmRun
2685 090E2 00          RTNYES
2686 090E4 85E         ST=1   NoCont          Set NoCont flag
2687 090E7 71DE =STMBCL GOSUB  stmbfd
2688            * Buf length in A
2689 090EB 133         AD1EX
2690 090EE 101         R1=A                   Save Begin Dest
2691 090F1 8F00        GOSBVL =I/OCOL         Collapse buffer
           000
2692 090F8 111         A=R1
2693 090FB E0          A=A-B  A               Begin Source
2694 090FD 100         R0=A
2695 09100 1F49        D1=(5) =RVMEMS
           5F2
2696 09107 AC3         D=0    S
2697 0910A B47         D=D+1  S
2698 0910D 8C00        GOLONG =FORUPD
           00
2699            *
2700            ********************************************************************
2701            ********************************************************************
2702            **
```

```
2703            ** Name:    KBRTCK  -  Keyboard return check
2704            **
2705            ** Category: EXCUTL
2706            **
2707            ** Purpose : If a statement is to return to statement buffer,
2708            **           check if the statement buffer still there.
2709            **
2710            **           Clear PgmRun and PRGM annunciator
2711            **           Clear SUSP annunciator
2712            **           If CNTADR is non-zero
2713            **              Re-suspended prior suspended program
2714            **           If CNTADR is zero and the Stmt Buffer is empty
2715            **              Zero program info, issue zero program poll
2716            **
2717            ** Entry : P = 0
2718            **
2719            ** Exit : If statement buffer empty, exits to EXITRN which
2720            **           returns to MAINLP through BASIC Loop Exit
2721            **
2722            ** Calls : I/OFND, NOPRGM, CNTCK2, SETSU1, CPL#10, SAVEL#,
2723            **         CLSUSP, ZERPGM
2724            **
2725            ** Uses : A-D, D0,D1, ALRM(+36), PNDALM(+1), CURRL
2726            **
2727            ** Stk lvls : 3
2728            **
2729            ** NOTE:  This is the maximum ALLOWED usage
2730            **        User Defined functions from the keyboard
2731            **        call this and their max usage is 4
2732            **
2733            ** Note:
2734            **
2735            **   GOSUB, CALL, DEF save the pending CNTADR
2736            **   RETURN, ENDSUB, ENDDEF restore the CNTADR
2737            **   If the CNTADR is non zero
2738            **      A program was suspended prior to invoking the
2739            **        GOSUB, CALL, DEF
2740            **      To restore this state:
2741            **        Set the SUSP annunciator
2742            **        CNTADR points at EOL or "@" before stmt to continue at
2743            **        Subsequent line# computution:
2744            **          Computes line number for statement of CNTADR
2745            **        Update current line
2746            **     If Continue address = 0 and the statement buffer is empty
2747            **        Program info must be cleared
2748            **        A subprogram, user def function or program was
2749            **        invoked from the keyboard, interrupted, then the
2750            **        invoking statement was destroyed but a subsequent
2751            **        keyboard entry.
2752            **
2753            ** History:
2754            **
2755            ** Date      Programmer         Modification
2756            ** --------- -----------        -------------------------------
2757            ** 03/07/83  J.P.               Added NOPRGM call,if SST,CLRSTK
```

```
2758            ** 03/07/83  J.P.            Change MAINLP exit to BSCEXT
2759            ** 03/18/83  J.P.            Alway clear addresses,memory
2760            ** 04/04/83  J.P.            Don't collapse stacks,just zero
2761            ** 04/25/83  J.P.            Exit through EXITRN not BSCEXT
2762            ** 05/13/83  J.P.            Updated Uses and Stk lvl document
2763            ** 05/27/83  J.P.            If CNTADR#0; SUSP program
2764            ** 06/06/83  J.P.            If CNTADR=0 and Stmt Buffer= 0
2765            **                              Zero program information
2766            ** 06/17/83  J.P.            Don't use PCADDR to update CURRL
2767            **                           CURRL @ CNTADR now !!!!!
2768            ** 06/17/83  J.P.            Change stack usage from 4 to 3
2769            **
2770            ********************************************************************
2771            ********************************************************************
2772            *
2773 09113 8E00 =KBRTCK GOSUBL =NOPRGM        Clear PgmRun and PRGM Annunciator
          00
2774 09119 8E00        GOSUBL =CLSUSP        Clear SUSP annunciator
          00
2775 0911F 8E00        GOSUBL =CNTCK2        Read continue address
          00
2776 09125 432         GOC    KBRT20         CNTADR = 0
2777 09128 8E00        GOSUBL =SETSU1        Set SUSP; Incr. CNTADR into stmt
          00
2778 0912E 8E00        GOSUBL =CPL#10        Compute line# @ DO
          00
2779 09134 8E00        GOSUBL =SAVEL#        Update CURRL
          00
2780 0913A 7E7E  KBRT10 GOSUB  stmbfd         Is Statement Buffer empty ?
2781 0913E 93C         ?A#0   X              NOT EMPTY ?
2782 09141 00          RTNYES
2783 09143 8C00  KBRT15 GOLONG =EXITRN        CLRST;to BSCEXT in BASIC loop
          00
2784            *
2785            * If CNTADR = 0 and Statment Buffer empty
2786            *    Zero program information
2787            *    Exit BASIC loop
2788            * else
2789            *    Return to statement buffer
2790            *
2791 09149 7F6E  KBRT20 GOSUB  stmbfd         Check statement buffer
2792 0914D 93C         ?A#0   X              Not empty ?
2793 09150 00          RTNYES
2794 09152 8E00        GOSUBL =ZERPGM        Zero program information
          00
2795 09158 5AE         GONC   KBRT15         B.E.T.
2796            *
2797            ********************************************************************
2798            ********************************************************************
2799            **
2800            ** Name:(S) SCOPCK  -  Scope check
2801            **
2802            ** Category: EXCUTL
2803            **
2804            ** Purpose: Verifies if an address is in current program scope
```

```
2805                **
2806                ** Entry : A(A)= ADDRESS TO BE VERIFIED
2807                **
2808                ** Exit:
2809                **       A is preserved from entry
2810                **       Carry clear - Address in current program scope
2811                **       Carry set   - Address out of current program scope
2812                **
2813                ** Calls: none
2814                **
2815                ** Uses: C(A),D0
2816                **
2817                ** Stk lvls: +0
2818                **
2819                ********************************************************************
2820                ********************************************************************
2821                *
2822 0915B 1B26 =SCOPCK D0=(5) =PRGMST
           5F2
2823 09162 146          C=DAT0 A
2824 09165 8B2          ?A<C   A
2825 09168 00           RTNYES
2826 0916A 164          D0=D0+ (PRGMEN)-(PRGMST)
2827 0916D 146          C=DAT0 A
2828 09170 8BE          ?A>=C  A
2829 09173 00           RTNYES
2830 09175 03           RTNCC
2831                *
```

```
2832                    EJECT
2833                ********************************************************************
2834                ********************************************************************
2835                **
2836                ** Name:    IF      -  IF statement execute
2837                **
2838                ** Category:   STEXEC
2839                **
2840                ** Purpose:    EXECUTES IF STATEMENT
2841                **
2842                ** Entry:      D0 POINTS TO EXPRESSION
2843                **
2844                ** Exit:       If tTHEN or tELSE followed by tEXTIF
2845                **                exit via NXTST5 with D0 at tEXTIF
2846                **             Otherwise exits via GOTO
2847                **                with D0 at line# or label token
2848                **
2849                **             Error exits only if expression is complex
2850                **
2851                ** Calls:    EXPEXC, EOLSCN, POP1N+, FINDA, LINSKP, UPDPCC
2852                **
2853                ** Uses:     A-D, R0-R3, D1,D0, S0-S11, function scratch
2854                **
2855                ** Stack lvls: 6
2856                **
2857                ** History:
2858                **
2859                **    Date     Programmer   Modifications
2860                **  --------   ----------   -------------
2861                **  07/04/82   S.W.         Added documentation
2862                **  07/28/82   S.W.         Modified to make PCADDR
2863                **                           point prior to extended stmt
2864                **                           immediately after THEN/ELSE
2865                **  05/04/83   S.W.         Replaced call to POP1N with
2866                **                           call to POP1N+; also no longer
2867                **                           error on complex
2868                **
2869                ********************************************************************
2870                ********************************************************************
2871                *
2872 09177 0000          REL(5) =IFDC
            0
2873 0917C 0000          REL(5) =IFP
            0
2874 09181 8E00 =IF      GOSUBL =EXPEXC
            00
2875 09187 8E00          GOSUBL =POP1N+
            00
2876 0918D 04            SETHEX
2877 0918F 2E            P=     14
2878 09191 5A0           GONC   IFXQ03          Real ?
2879                * Number is complex - Test real & imaginary parts
2880 09194 91C           ?A#0   WP              Real part nonzero =>
2881 09197 C4            GOYES  IFXQ10           execute THEN portion
2882 09199 110           A=R0                   Imaginary part#0 => same
```

```
2883 0919C 91C   IFXQ03 ?A#0   WP
2884 0919F 44           GOYES  IFXQ10          IF NON-ZERO,RUN 'THEN'
2885            * ELSE PATH
2886 091A1 7A58         GOSUB  LNSKP-
2887 091A5 14A   IFXQ05 A=DAT0 B
2888 091A8 20           P=     0
2889 091AA 908          ?A=0   P               At tEOL?
2890 091AD 12           GOYES  IFXQ07
2891 091AF 163          DO=DO+ 4               Step over t@ & stmt len
2892 091B2 14A          A=DAT0 B
2893 091B5 181          DO=DO- 2               Point to stmt length
2894 091B8 3100         LC(2)  =tELSE
2895 091BC 962          ?A=C   B               tELSE ?
2896 091BF 51           GOYES  IFXQ08
2897 091C1 8EE3         GOSUBL LINSKP          Skip to next stmt
           8F
2898 091C7 5DD          GONC   IFXQ05          (B.E.T.)
2899            *
2900 091CA 79D8 =ELSE   GOSUB  EOLSCN          Skip to tEOL
2901 091CE 8C61  IFXQ07 GOLONG NXTST5
           7F
2902            *
2903            * Set PCADDR here to fix bug :
2904            * > IF 0 THEN <extended command> ELSE <line# or label>
2905            *
2906            * In the case of extended command after ELSE, PCADDR will
2907            *  get written over anyway
2908            *
2909 091D4 136   IFXQ08 CDOEX
2910 091D7 8E00         GOSUBL =UPDPCC         Update PCADDR from C
           00
2911 091DD 134          DO=C
2912 091E0 161          DO=DO+ 2               Step over StmtLen
2913 091E3 8E00  IFXQ10 GOSUBL =FNDDO+         Step over tTHEN/tELSE
           00
2914 091E9 00           CON(2) =tEXTIF         Is this an extended IF ?
2915 091EB 3EF          REL(3) IFXQ07
2916 091EE 00           CON(2) =t!
2917 091F0 ADF          REL(3) ELSE
2918 091F3 00           CON(2) 0
2919            *
2920 091F5 8C00         GOLONG =GOTO
           00
2921            *
2922            *
2923            ****************************************************************
2924            ****************************************************************
2925            **
2926            ** Name:(S) COLLAP  -  Collapse Math Stack
2927            **
2928            ** Category:   MTHSTK
2929            **
2930            ** Purpose:
2931            **      Collapses math stack
2932            **
```

```
2933                ** Entry:
2934                **
2935                ** Exit:
2936                **      D1 = MTHSTK
2937                **      C(A)= new value of MTHSTK pointer
2938                **      Carry clear
2939                **
2940                ** Calls:      none
2941                **
2942                ** Uses.......
2943                **      C(A),D1
2944                **
2945                ** Stk lvls:   0
2946                **
2947                ** History:
2948                **
2949                **    Date     Programmer              Modification
2950                **  --------   ----------   ---------------------------------
2951                **  06/25/82   S.W.         Created utility
2952                **
2953                ********************************************************************
2954                ********************************************************************
2955                *
2956 091FB 1FE9 =COLLAP D1=(5) =FORSTK
           5F2
2957 09202 147          C=DAT1 A
2958 09205 1C4          D1=D1- 5
2959 09208 145          DAT1=C A
2960 0920B 01           RTN
2961                *
```

```
2962                    STITLE KEYDOWN Function
2963                ****************************************************************
2964                ****************************************************************
2965                **
2966                ** Name:    KEYDWN  -  KEYDOWN function execution
2967                **
2968                ** Category:   FNEXEC
2969                **
2970                ** Purpose:
2971                **      Executes KEYDOWN function
2972                **
2973                ** Entry:
2974                **      P       =  0
2975                **
2976                ** Exit:
2977                **      P       =  0
2978                **
2979                ** Calls:      KEYSCN, GTKYC*, POPMTH
2980                **
2981                ** History:
2982                **
2983                **    Date      Programmer              Modification
2984                **  --------   ----------   ---------------------------------
2985                **  09/12/83   B.S.         Updated documentation
2986                **
2987                ****************************************************************
2988                ****************************************************************
2989 0920D 401             NIBHEX 401             Argument range [0,1]
2990 09210 136     =KEYDWN CDOEX
2991 09213 10A             R2=C                   Save D0 in R2
2992 09216 8F00            GOSBVL =KEYSCN         Update key down map
           000
2993 0921D 11A             C=R2
2994 09220 94A             ?C=0    S              Passed a parameter?
2995 09223 D6              GOYES   KEYD30         No, then check if any keys down
2996 09225 137             CD1EX
2997 09228 108             R0=C                   Save D1 in R0
2998 0922B 755B            GOSUB   GTKYC*          Get key code
2999 0922F 3183            LC(2)   56
3000 09233 9E1             ?B>C    B              Is it a primary key?
3001 09236 76              GOYES   RNGERR         No, then argument out of range
3002 09238 110             A=R0
3003 0923B 7740            GOSUB   popmt+         Pop argument off stack
3004 0923F CD              B=B-1   A
3005 09241 D0              A=0     A
3006 09243 4B2             GOC     KEYD20         Return false if null string passed
3007 09246 D2              C=0     A
3008 09248 308             LCHEX   8
3009 0924B DA              A=C     A
3010 0924D 31E0            LC(2)   14
3011 09251 E1      KEYD10  B=B-C   A
3012 09253 480             GOC     KEYD15
3013 09256 81C             ASRB
3014 09259 57F             GONC    KEYD10         (B.E.T.)
3015                *-
```

```
3016                *-
3017 0925C 136     KEYD15 CDOEX
3018 0925F E9             C=C-B  A
3019 09261 134            DO=C
3020 09264 18E            DO=DO- 15
3021 09267 15E0           C=DATO 1
3022 0926B 0E06           A=A&C  P
3023 0926F B98     KEYD20 A=-A   WP
3024 09272 AF2            C=0    W
3025 09275 570            GONC   KEYD25
3026 09278 2E             P=     14
3027 0927A 301            LCHEX  1
3028 0927D 112     KEYD25 A=R2
3029 09280 8C00           GOLONG =FNRTN2
           00
3030                *-
3031                *-
3032 09286 131     popmt+ D1=A
3033 09289 8D00    popmth GOVLNG =POPMTH
           000
3034                *-
3035                *-
3036 09290         KEYD30
3037 09290 18D            DO=DO- (DISINT)-(KEYSAV)
3038 09293 2D             P=     13              14 key rows
3039 09295 1521           A=DATO WP              Read 14 columns
3040 09299 65DF           GOTO   KEYD20
3041                *-
3042                *-
3043 0929D 642B    RNGERR GOTO   GTINAR          Invalid argument
3044                *-
3045                *-
3046 092A1 55E6    UNASS" NIBASC \Unassign\
           1637
           3796
           76E6
3047 092B1 5646           NIBASC \ed\
3048 092B5 FF             NIBHEX FF
3049                *-
3050                *-
```

```
3051                    STITLE PUT Statement
3052                ****************************************************************
3053                ****************************************************************
3054                **
3055                ** Name:    PUT     -  PUT statement execution
3056                **
3057                ** Category:   STEXEC
3058                **
3059                ** Purpose:
3060                **      Executes PUT statement
3061                **
3062                ** Entry:
3063                **      P      =  0
3064                **      DO points past PUT token
3065                **
3066                ** Exit:
3067                **      Exits through NXTSTM
3068                **
3069                ** Calls:      GTKYCD, KEYFND, POPMTH, KEYTYP, STKCHR
3070                **
3071                ** History:
3072                **
3073                **    Date     Programmer              Modification
3074                **  --------   ----------   ----------------------------------
3075                **  09/12/83   B.S.         Updated documentation
3076                **
3077                ****************************************************************
3078                ****************************************************************
3079 092B7 0000            REL(5) =FIXDC
           0
3080 092BC 0000            REL(5) =STRNGP        Parse(Allow string expr)
           0
3081 092C1 7DCA =PUT       GOSUB  GTKYCD         Get key code by executing expr
3082 092C5 3434            LC(5)  =KEYPTR
           4F2
3083 092CC 135             D1=C
3084 092CF D0              A=0    A
3085 092D1 808F            INTOFF
3086 092D5 1530            A=DAT1 P              Read key pointer
3087 092D9 C2              C=C+A  A
3088 092DB B04             A=A+1  P              Increment number of keys in buffer
3089 092DE 401             GOC    PUTK10         Too many? Yes, then forget it
3090 092E1 1510            DAT1=A P              Update pointer
3091 092E5 C2              C=C+A  A              Calculate position in buffer
3092 092E7 135             D1=C
3093 092EA D9              C=B    A
3094 092EC 14D             DAT1=C B              Write key into buffer
3095 092EF 8080  PUTK10    INTON
3096 092F3 661D            GOTO   nxtstm         Return for next statement
3097                *-
```

```
3098                    STITLE KEYDEF$ function execution
3099                ******************************************************************
3100                ******************************************************************
3101                **
3102                ** Name:    KEYDEF  -  KEYDEF$ function execution
3103                **
3104                ** Category:   FNEXEC
3105                **
3106                ** Purpose:
3107                **      Implements KEYDEF$ function
3108                **
3109                ** Entry:
3110                **      P =  0
3111                **      D1 is stack pointer
3112                **      D0 is PC
3113                **
3114                ** Exit:
3115                **      Exits through EXPR
3116                **
3117                ** Calls:      GTKYC*,KEYFND,GTINAR,POPMTH,BF2ST+,STKCHR,
3118                **             ADHEAD
3119                **
3120                ** History:
3121                **
3122                **    Date     Programmer             Modification
3123                **  --------   ----------   ----------------------------------
3124                **  10/25/83   B.S.         Added documentation
3125                **
3126                ******************************************************************
3127                ******************************************************************
3128 092F7 411             NIBHEX 411            Argument range [1,1]
3129 092FA 136  =KEYDEF    CD0EX
3130 092FD 108             R0=C                  Save D0 in R0
3131 09300 137             CD1EX
3132 09303 109             R1=C                  Save D1 in R1
3133 09306 7A7A            GOSUB  GTKYC*
3134 0930A 560             GONC   KEYDF0
3135 0930D 64BA            GOTO   GTINAR
3136 09311 73A9  KEYDF0    GOSUB  KEYFND         Find key definition?
3137 09315 D7              D=C    A              Save entry length in D(A)
3138 09317 1B49            D0=(5) =AVMEMS
           5F2
3139 0931E 146             C=DAT0 A              Read AVMEMS
3140 09321 DF              CDEX   A              Put in D(A), C(A)=Entry length
3141 09323 412             GOC    KEYDF1         Yes, then display definition
3142 09326 111             A=R1
3143 09329 795F            GOSUB  popmt+
3144 0932D 110             A=R0
3145 09330 130             D0=A                  Restore D0
3146 09333 341A            LC(5)  UNASS"
           290
3147 0933A 6000            GOTO   =bf2st+        Push string on stack and goto EXPR
3148                *-
3149
3150 0933E 8D00 =MOveu3 GOVLNG =MOVEU3           For several jumps...
```

```
            000
   3151
   3152             *-
   3153 09345 8F00  KEYDF1 GOSBVL =KEYTYP
            000
   3154 0934C 103          R3=A                    Added by NM 3 Nov 82
   3155 0934F 129          CR1EX                   R1=Def type char,C(A)=D1 save
   3156 09352 137          CD1EX                   Restore D1,C(A)=Defintion pointer
   3157 09355 134          D0=C                    D0 points to definition
   3158 09358 7D2F         GOSUB  popnth           Pop parameter off stack
   3159 0935C 137          CD1EX
   3160 0935F 135          D1=C                    Copy D1 to C (Stack pointer)
   3161 09362 129          CR1EX                   C(B)=Def type char,R1=Stack start
   3162 09365 8E00         GOSUBL =stkchr          Output def type char
            00
   3163 0936B 113          A=R3                    Recall def length+1
   3164 0936E CC    KEYDF2 A=A-1  A
   3165 09370 CC           A=A-1  A
   3166 09372 411          GOC    KEYDF3           Exit loop if all chars done
   3167 09375 14E          C=DAT0 B                Read char of definition
   3168 09378 161          D0=D0+ 2                Move pointer past char
   3169 0937B 8E00         GOSUBL =stkchr          Output char to stack
            00
   3170 09381 5CE          GONC   KEYDF2           (B.E.T.)
   3171             *-
   3172             *-
   3173 09384 110   KEYDF3 A=R0
   3174 09387 130          D0=A                    Restore D0
   3175 0938A 840          ST=0   0                Don't return from ADHEAD
   3176 0938D 8C00         GOLONG =Adhead          Add string header and jump to EXPR
            00
   3177 09393              END
```

```
 ARANGE  Ext                  -   683
 ARGERR  Ext                  -  1725
 ASNMNT  Ext                  -   128
 AVMEME  Abs  193945 #2F599   -    12   572   574  1145  1410  2026  2088
 AVMEMS  Abs  193940 #2F594   -    12  2695  3138
 Adhead  Ext                  -  3176
=BANG    Abs   35400 #08A48   -   905
 BSCXLP  Ext                  -   900
 C=MSTK  Abs   35312 #089F0   -   764   523   755
 CLSUSP  Ext                  -  2774
 CNTADR  Abs  194174 #2F67E   -    12  2636
 CNTCK2  Ext                  -  2775
=COLLAP  Abs   37371 #091FB   -  2956
 CONVUC  Ext                  -  1745
=CORUPT  Abs   36995 #09083   -  2587  2573  2601
 CPL#10  Ext                  -  2778
 CREATF  Ext                  -  1226
 CRLFSD  Ext                  -  1327
 DO=PCA  Ext                  -   243   807
 D1FSTK  Ext                  -   424   764
=DATA    Abs   35360 #08A20   -   886
 DATADC  Ext                  -   884
 DATP    Ext                  -   885
 DECHEX  Ext                  -  1825
 DISINT  Abs  193648 #2F470   -    12  3037
 DRANGE  Ext                  -  1819
 DT1=C   Abs   35304 #089E8   -   757   527
=ELSE    Abs   37322 #091CA   -  2900  2917
=EOLSCN  Abs   35495 #08AA7   -  1028   271  2900
=EOLSN5  Abs   35505 #08AB1   -  1031  1036
=EOLSN7  Abs   35513 #08AB9   -  1034
 ERRSUB  Abs  194179 #2F683   -    12  2548
 EXITRN  Ext                  -  2783
 EXPEX-  Ext                  -   751
 EXPEXC  Ext                  -   391  1105  1707  2874
 F-RO-1  Abs  194720 #2F8A0   -    12  1242
 FILEF   Ext                  -  1482
 FIXDC   Ext                  -  3079
 FNDDO+  Ext                  -  2913
 FNRTN2  Ext                  -  3029
=FOR     Abs   34394 #0865A   -    96
 FOR10   Abs   34443 #0868B   -   115    99
 FORDC   Ext                  -    94
 FORERR  Abs   34380 #0864C   -    92   118
 FORP    Ext                  -    95
 FORSTK  Abs  193950 #2F59E   -    12   402   574   693  2956
 FORUPD  Ext                  -  2698
 FORXQe  Abs   34376 #08648   -    90   131
 FPoll   Ext                  -  2513
 FRNLO4  Abs   34753 #087C1   -   276   269
 FRNLO5  Abs   34767 #087CF   -   282   280
 FRNLO7  Abs   34771 #087D3   -   284   301
 FRNL20  Abs   34842 #0881A   -   311   306
 FRNULL  Abs   34704 #08790   -   258   245
 FRTEST  Abs   35182 #0896E   -   636   633
```

```
 FRXQ30  Abs   34545 #086F1 -   169   145
 FRXQ35  Abs   34565 #08705 -   176   161
 FRXQ40  Abs   34568 #08708 -   177   164
 FRXQ45  Abs   34670 #0876E -   238   215
 FRXQ50  Abs   34674 #08772 -   243   213   236
 GETPR   Ext                -  1624
 GETST   Ext                -   127
 GOTO    Ext                -  2920
 GSBSTK  Abs  193955 #2F5A3 -    12  1926  2098
 GTINAR  Abs   36290 #08DC2 -  1725  1740  1753  1757  1760  1766  1785  3043
                            3135
 GTKY05  Abs   36296 #08DC8 -  1727  1715
 GTKY10  Abs   36367 #08E0F -  1755  1720  1749
 GTKY15  Abs   36422 #08E46 -  1770  1774
 GTKY17  Abs   36454 #08E66 -  1785  1809  1820  1827  1834
 GTKY25  Abs   36444 #08E5C -  1780  1783
 GTKY30  Abs   36458 #08E6A -  1787  1781
 GTKY34  Abs   36463 #08E6F -  1789  1804
 GTKY35  Abs   36465 #08E71 -  1790  1788  1802
 GTKY40  Abs   36469 #08E75 -  1792  1798
 GTKY45  Abs   36473 #08E79 -  1796  1778
 GTKY50  Abs   36492 #08E8C -  1806  1700
 GTKY55  Abs   36511 #08E9F -  1816  1823
 GTKY60  Abs   36526 #08EAE -  1820  1852  1855
 GTKY65  Abs   36571 #08EDB -  1836  1839
 GTKY70  Abs   36581 #08EE5 -  1841  1837
 GTKY80  Abs   36599 #08EF7 -  1849  1792  1845
 GTKYC*  Abs   36228 #08D84 -  1702  2998  3133
=GTKYC+  Abs   36251 #08D9B -  1709  1705
=GTKYCD  Abs   36242 #08D92 -  1707  1086  3081
 GTKYNM  Abs   36224 #08D80 -  1700  1736
 I/OCOL  Ext                -  2691
=IF      Abs   37249 #09181 -  2874
 IFDC    Ext                -  2872
 IFP     Ext                -  2873
 IFXQ03  Abs   37276 #0919C -  2883  2878
 IFXQ05  Abs   37285 #091A5 -  2887  2898
 IFXQ07  Abs   37326 #091CE -  2901  2890  2915
 IFXQ08  Abs   37332 #091D4 -  2909  2896
 IFXQ10  Abs   37347 #091E3 -  2913  2881  2884
=IMAGE   Abs   35400 #08A48 -   903
 INDX02  Abs   35219 #08993 -   691   685
 INDX04  Abs   35243 #089AB -   698   709
 INDX07  Abs   35277 #089CD -   711   700
 INDX08  Abs   35279 #089CF -   712   706
 INDXSN  Abs   35194 #0897A -   681    96   377
 INVRES  Ext                -   224
 KBRT10  Abs   37178 #0913A -  2780
 KBRT15  Abs   37187 #09143 -  2783  2795
 KBRT20  Abs   37193 #09149 -  2791  2776
=KBRTCK  Abs   37139 #09113 -  2773  2638
=KEY     Abs   35533 #08ACD -  1086
 KEY42   Abs   35564 #08AEC -  1100  1093
 KEY45   Abs   35623 #08B27 -  1118  1116
 KEY47   Abs   35652 #08B44 -  1128  1123
```

KEY50	Abs	35655	#08B47	-	1130	1126			
KEYCOD	Ext			-	1767				
KEYD10	Abs	37457	#09251	-	3011	3014			
KEYD15	Abs	37468	#0925C	-	3017	3012			
KEYD20	Abs	37487	#0926F	-	3023	3006	3040		
KEYD25	Abs	37501	#0927D	-	3028	3025			
KEYD30	Abs	37520	#09290	-	3036	2995			
KEYDC	Ext			-	1084				
=KEYDEF	Abs	37626	#092FA	-	3129				
=KEYDEL	Abs	36140	#08D2C	-	1562	1095			
KEYDF0	Abs	37649	#09311	-	3136	3134			
KEYDF1	Abs	37701	#09345	-	3153	3141			
KEYDF2	Abs	37742	#0936E	-	3164	3170			
KEYDF3	Abs	37764	#09384	-	3173	3166			
=KEYDWN	Abs	37392	#09210	-	2990				
=KEYFND	Abs	36024	#08CB8	-	1479	1211	1562	3136	
=KEYMRG	Abs	35727	#08B8F	-	1202	1135			
KEYP	Ext			-	1085				
KEYPTR	Abs	193603	#2F443	-	12	3082			
KEYSAV	Abs	193634	#2F462	-	12	3037			
KEYSCN	Ext			-	2992				
KEYTYP	Ext			-	3153				
=KMEMCK	Abs	35915	#08C4B	-	1311	1223	1258		
KMRG30	Abs	35828	#08BF4	-	1241	1216	1219		
KMRG55	Abs	35872	#08C20	-	1258	1249			
KMRG85	Abs	35899	#08C3B	-	1267	1255			
KYD30	Abs	36177	#08D51	-	1575	1254			
KYFD10	Abs	36090	#08CFA	-	1504	1517			
KYFD20	Abs	36127	#08D1F	-	1519	1514			
KYFD30	Abs	36134	#08D26	-	1523	1483			
KYFD40	Abs	36138	#08D2A	-	1525	1506			
=KYFND+	Abs	36052	#08CD4	-	1489				
=KYMRG+	Abs	35746	#08BA2	-	1209				
KYPRC2	Abs	36215	#08D77	-	1630	1627			
KYPRCK	Abs	36190	#08D5E	-	1619	1244	1564		
=LABEL	Abs	35364	#08A24	-	893				
LABL10	Abs	35367	#08A27	-	894	897			
LAKEYS	Ext			-	1229	1479			
=LINSKP	Abs	35333	#08A05	-	809	247	285	1035	2897
=LNSKP-	Abs	35327	#089FF	-	807	1030	2886		
MEMBER	Ext			-	1765				
MEMCKL	Ext			-	117				
MEMCL+	Ext			-	1311	1401			
MFERR	Ext			-	1314	1628	2508	2588	
MOVED2	Ext			-	1263				
MOVEU1	Ext			-	1415				
MOVEU3	Ext			-	3150				
MOVEUA	Ext			-	1575				
=MOVNXT	Abs	35719	#08B87	-	1155				
MOved3	Ext			-	2036				
=MOveu3	Abs	37694	#0933E	-	3150	1155			
=NEXT	Abs	34892	#0884C	-	377				
NEXT10	Abs	34925	#0886D	-	391	370			
NEXT50	Abs	35040	#088E0	-	444	2639			
NEXT55	Abs	35043	#088E3	-	445	434			

```
 NEXTKB  Abs   34846 #0881E -   360   389
 NOPRGM  Ext                 - 2773
 NXTCK7  Abs   35118 #0892E -   568   238   450
 NXTCK9  Abs   35124 #08934 -   571   124   133   174   251   383
 NXTDC   Ext                 -   375
 NXTDNE  Abs   35053 #088ED -   450   427
 NXTE32  Abs   34874 #0883A -   372   366   369   378   387
 NXTP    Ext                 -   376
 NXTSN7  Abs   34799 #087EF -   296   292
=NXTST1  Abs   35403 #08A4B -   907
=NXTST2  Abs   35416 #08A58 -   912   311
 NXTST3  Abs   35429 #08A65 -   917   447   899
 NXTST5  Abs   35046 #088E6 -   446   252   440  2901
=NXTSTM  Abs   35400 #08A48 -   906   451   886  1096  1156  2509
 NoCont  Abs      14 #0000E -    11  2686
 PCADDR  Abs  194169 #2F679 -    12   515   907
=POP     Abs   36815 #08FCF - 2478
 POP1N+  Ext                 - 2875
 POP1R   Ext                 -   393   752
 POP1S   Ext                 - 1709
=POPGSB  Abs   36670 #08F3E - 2014  2559
 POPMTH  Ext                 - 3033
 POPSH   Abs   35283 #089D3 -   750   137   146
=POPSTK  Abs   36693 #08F55 - 2026   109   192
=POPUPD  Abs   36670 #08F3E - 2015  2634
 PRGMEN  Abs  193895 #2F567 -    12   262  2826
 PRGMST  Abs  193890 #2F562 -    12  2822  2826
=PSHGSB  Abs   36627 #08F13 - 1918
=PSHMCR  Abs   36619 #08F0B - 1914
=PSHSTK  Abs   35967 #08C7F - 1393  1927
=PSHSTL  Abs   35973 #08C85 - 1400
=PSHUPD  Abs   36621 #08F0D - 1916
=PTRAD1  Abs   36739 #08F83 - 2088  1427
=PTRAD2  Abs   36746 #08F8A - 2089  2094
=PUT     Abs   37569 #092C1 - 3081
 PUTK10  Abs   37615 #092EF - 3095  3089
 PgmRun  Abs      13 #0000D -    11  2623  2684
 RACTMI  Ext                 - 2602
 RDATTY  Ext                 - 1098
=REM     Abs   35400 #08A48 -   904
 REMDC   Ext                 -   901
 REMP    Ext                 -   902
 RETRNp  Ext                 - 2486
=RETURN  Abs   36832 #08FE0 - 2487
 REV$    Ext                 - 1109
 RFAD++  Ext                 - 1265
 RFAD--  Ext                 - 1578
 RNGERR  Abs   37533 #0929D - 3043  3001
 RSTD1   Ext                 - 1270
 RSTST   Ext                 -   258
 RTNCC   Ext                 - 2477
 RTNERR  Abs   36857 #08FF9 - 2504  2515  2526
 RTNSTK  Abs   36766 #08F9E - 2098  2015  2495
 RTNX10  Abs   36850 #08FF2 - 2495  2479  2490
 RTNX20  Abs   36878 #0900E - 2513  2534
```

```
 RTNX30  Abs   36890 #0901A -  2522  2496
 RTNX40  Abs   36924 #0903C -  2548  2532
 RTNX45  Abs   36944 #09050 -  2559  2551
 RTNX50  Abs   36978 #09072 -  2582  2571
 RTNX60  Abs   37003 #0908B -  2596  2583
 RTNX70  Abs   37025 #090A1 -  2617  2585
 RTNX75  Abs   37050 #090BA -  2623  2621
 RTNX80  Abs   37057 #090C1 -  2633  2597
 RTNX90  Abs   37081 #090D9 -  2639  2624
 RUNRTN  Ext                -   917
 S-R0-0  Abs  194673 #2F871 -   12   399  1101  1130
 S-R1-2  Abs  194699 #2F88B -   12   520
 SAVEL#  Ext                -  2779
=SCOPCK  Abs   37211 #0915B -  2822  2584
 SETSU1  Ext                -  2777
=SNcrlf  Abs   35954 #08C72 -  1325  2506  2574  2586  2633
=STMBCL  Abs   37095 #090E7 -  2687
=STMBUF  Abs   37087 #090DF -  2684  2488
 STOPDC  Ext                -  2476  2485
 STORE   Ext                -   529
 STRNGP  Ext                -  3080
=TKSCN+  Abs   35435 #08A6B -   965
 TKSCN2  Abs   35438 #08A6E -   966   989
=TKSCN4  Abs   35441 #08A71 -   967   986
=TKSCN7  Abs   35481 #08A99 -   982   291
 TRCVAR  Abs   35061 #088F5 -   510   230   417
 TRCVR5  Abs   35097 #08919 -   523   514
 TRFLCK  Ext                -  1323
 TRFROM  Ext                -  1319
 TRTO+   Ext                -   443
 TRTO-   Ext                -   308
 TRTOr   Ext                -  2622
 TST12A  Ext                -   637
 UNASS"  Abs   37537 #092A1 -  3046  3146
 UNORD   Abs   34629 #08745 -   220   209
 UNSTEP  Abs       2 #00002 -   155   163   176   208
 UPDFCL  Ext                -  1268
 UPDPCA  Ext                -  2618
 UPDPCC  Ext                -  2910
 XQBLK?  Abs   35148 #0894C -   623   211   426
 XQEND   Abs   35190 #08976 -   638   630
 ZERPGM  Ext                -  2794
 bSTMT   Abs    2049 #00801 -    11  2109
 bf2st+  Ext                -  3147
 eFwoNX  Ext                -   293
 eMMCOR  Ext                -  2587
 eNXwoF  Ext                -   372
 eRwoGS  Ext                -  2507
 eUNORC  Ext                -   223
 fKEY    Abs   57868 #0E20C -    11  1232
 i/ofnd  Ext                -  2110
 lDATEh  Abs       6 #00006 -    11  1234
 lFLAGh  Abs       2 #00002 -    11  1234
 lFLENh  Abs       5 #00005 -    11  1235
 lFNAMh  Abs      16 #00010 -    11  1230
```

```
 lFTYPh  Abs      4 #00004 -   11  1233  1234
 lTIMEh  Abs      4 #00004 -   11  1234
 mferr   Abs  35924 #08C54 - 1314    92   294   373  1403
 nxtstm  Abs  36874 #0900A - 2509  2505  2540  2561  3096
 oFLENh  Abs     32 #00020 -   11  1221  1490
 oFTYPh  Abs     16 #00010 -   11  1231
 oKYsod  Abs      5 #00005 -   11  1221  1493  1565
 pRTNTp  Abs     58 #0003A -   11  2514
 popmt+  Abs  37510 #09286 - 3032  3003  3143
 popmth  Abs  37513 #09289 - 3033  3158
 rdatty  Abs  35557 #08AE5 - 1098    90  1108
=rstd1   Abs  35908 #08C44 - 1270
 rtncc   Abs  35348 #08A14 -  815   969
 sENDx   Abs      1 #00001 -   11   446   906
 sRETRN  Abs      0 #00000 -   11  2478  2487  2504  2539  2560
 stkchr  Ext                - 3162  3169
 stmbfd  Abs  36796 #08FBC - 2109   361  2687  2780  2791
 t!      Ext                - 2916
 tCOMMA  Ext                - 1091
 tELSE   Ext                - 2894
 tEXTIF  Ext                - 2914
 tNEXT   Ext                -  287
 tSTEP   Ext                -  143
 trcfr1  Abs  35940 #08C64 - 1320   309
 trcfrm  Abs  35928 #08C58 - 1317   307   442  2491
 trflck  Abs  35948 #08C6C - 1323   305   439  1325  2489  2620
 uAD12   Ext                -  409
 utst12  Abs  35184 #08970 -  637   160
```

Input Parameters

  Source file name is SG&EXC::MS

  Listing file name is SG/EXC:TI:ML::-1

  Object file name is SG%EXC:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *
 2            *    TTTTT  III   &     EEEEE  RRRR   DDDD
 3            *      T     I   & &    E      R   R  D  D
 4            *      T     I   & &    E      R   R  D  D
 5            *      T     I    &     EEEE   RRRR   D  D
 6            *      T     I   & & &  E      R R    D  D
 7            *      T     I   &  &   E      R  R   D  D
 8            *      T    III   && &  EEEEE  R   R  DDDD
 9            *
10                    TITLE  Error Message Driver <831216.1627>
11 09393              ABS    #9393
12            *
13                    RDSYMB TIXEQU::MS
14                    RDSYMB SBXRAM::MS
15            *
16            ************************************************************
17             InhEOL EQU    4              !!!! Same as SB&IO !!!
18             cntr?  EQU    (InhEOL)+1     Flag for DSPCNA.
19             width? EQU    (InhEOL)+2     Flag for DSPCNA.
20            *
21            *                       Entry options, stored in C(12):
22             delay  EQU    0              Inhibit delay.
23             parse  EQU    1              Parse error.
24             warn   EQU    2              Warning.
25             beep   EQU    3              Beep.
26            ********************************************************************
27            ********************************************************************
28            **
29            ** Name:(S) pWARN   -  Warning poll
30            **
31            ** Category:   POLL
32            **
33            ** Type:       POLL
34            **
35            ** Purpose:
36            **      Alert LEX files that a warning is about to go out.
37            **
38            ** Should poll be "Handled" (return with XM=0)?:
39            **      Only if you want the message to be entirely suppressed.
40            **      Most applications will "handle" the poll without
41            **      setting XM=0 (see below).
42            **
43            ** Meaning of "Handling" Poll (what does code do if XM=0?):
44            **      It's up to you.  For instance, a LEX file might want to
45            **      intercept all warnings and errors to write them to a
46            **      file; in this case, do your thing and return with XM=0
47            **      so that the message is suppressed.
48            **
49            ** Entry conditions for handler (registers, ST, RAM, etc.):
50            **      B[A] = Poll number.
51            **      HEX mode.
52            **      P=0.
53            **      A(A)=0 if Quiet (flag -1) is to be checked,
54            **           else A(A)=FFFFF if Quiet is not to be checked.
55            **
```

```
56          **          F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0
57          **     R0: |  |     |                          |     |     |
58          **         ^     ^                             ^     ^
59          **         |     |                             |     |
60          **         |  control codes for text insertion |     |
61          **         + Entry P value (see MFWRN)         |     |
62          **                              LEX ID of message     |
63          **                                          message number
64          **
65          **
66          ** Normal exit conditions from handler if handled (ST, RAM,
67          ** registers, etc.):
68          **      Carry clear.
69          **      HEX mode.
70          **      XM=0.
71          **      P=0.
72          **      no other requirements -- the message will be suppressed
73          **
74          ** Normal exit conditions from handler if not handled (ST, RAM,
75          ** registers, etc.):
76          **      Carry clear.
77          **      HEX mode.
78          **      XM=1.
79          **      P=0.
80          **      R0 can be changed as needed to adjust msg (see MFWRN)
81          **
82          ** Error exit conditions from handler:
83          **      Carry set.
84          **      HEX mode.
85          **      C[0-3] = Error number.
86          **      P= value to select options in MFERR*
87          **
88          ** Available subroutine levels:
89          **      6
90          **
91          ** What registers/RAM may be used if handled?:
92          **      A,B,C,D,D0,D1,P,R0
93          **      (Not available: R1,R2,R3,R4,ST, scratch RAM)
94          **
95          ** What registers/RAM may be used if not handled?:
96          **      A,B,C,D,D0,D1,P,R0 (change R0 only to affect msg)
97          **      (Not available: R1,R3,R4,ST, scratch RAM.
98          **        R2 unavailable except for rare cases when insertion
99          **        text is being passed to the msg routines.)
100         **
101         ** What registers/RAM may be used if error exit (POLL only)?:
102         **      A,B,C,D,D0,D1,P,R0
103         **      (Not available: R1,R2,R3,R4,ST, scratch RAM)
104         **
105         ** NOTE:
106         **      The pWARN poll (and other message polls) are usually
107         **      "handled" without setting XM=0.  This is to allow
108         **      all LEX files to get a chance to intercept the poll.
109         **
110         **      A LEX file which intercepts the poll has essentially
```

```
111             **      four choices:
112             **        1) Abort the warning message, continue executing
113             **           or whatever else it wants to do (including
114             **           jumping instead to the error routine).
115             **        2) Change the values in R0 to cause a different
116             **           warning to be reported, or to cause different
117             **           entry conditions as selected by the value in
118             **           R0(S), or to cause different text insertion
119             **           by changing the values in R2 (text insertion
120             **           applies only to certain rare messages). Then
121             **           allow the poll to return to the warning routine
122             **           with XM=1.
123             **        3) Simply clear XM ("poll handled"). This causes
124             **           the message to be suppressed; message driver
125             **           returns immediately (without setting ERRN, etc.)
126             **        4) If error is generated by poll handler, set carry
127             **           and load error number in C(3-0). This will
128             **           cause a jump to BSERR with the new error number.
129             **
130             **
131             ** Envisioned application(s):
132             **    A) Foreign Language Translators: if the warning message
133             **       number is from the appropriate LEX file, the message
134             **       number in R0 is adjusted to generate the translator's
135             **       message. (If a type {5} building block is included
136             **       in the message, this will have to be adjusted through
137             **       a nested pTRANS poll, too. See IDS volume I, chapter
138             **       "Message Handling".) Set XM=1 and return.
139             **    B) Say a LEX file intercepts all warnings, writes the
140             **       message number (ERRN) and line number (ERRL) to a
141             **       file, and suppresses the display of the warning.
142             **       When intercepting this poll, it would do the
143             **       necessary processing and return with XM=0.
144             **    C) Say another operating system will not allow any
145             **       warnings to be issued, only errors. It could
146             **       intercept the pWARN poll and jump directly to
147             **       BSERR so that the warning is converted into an error.
148             **    D) An automated card puller (!?!?) might trap the
149             **       appropriate card reader messages and use them as
150             **       prompts when to insert and pull cards.
151             **
152             **
153             ** History:
154             **
155             **    Date     Programmer            Modification
156             **  --------   ----------   ------------------------------------
157             **  10/05/82   MB           Documentation
158             **  01/27/83   MB           Added "poll handled" suppress
159             **
160             ********************************************************************
161             ********************************************************************
```

```
                EJECT
*****************************************************************
*****************************************************************
**
** Name:(S) pERROR  -  Error poll
**
** Category:   POLL
**
** Type:       POLL
**
** Purpose:
**      Alert LEX files that an error is about to go out.
**
** Should poll be "Handled" (return with XM=0)?:
**      Only if you want the message to be entirely suppressed.
**      Most applications will "handle" the poll without
**      setting XM=0 (see below).
**
** Meaning of "Handling" Poll (what does code do if XM=0?):
**      It's up to you.  For instance, a LEX file might want to
**      intercept all errors and warnings to write them to a
**      file; in this case, do your thing and return with XM=0
**      so that the message is suppressed.
**
** Entry conditions for handler (registers, ST, RAM, etc.):
**      B[A] = Poll number.
**      HEX mode.
**      P=0.
**
**           F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0
**      R0: |  |     |                          |     |     |
**          ^    ^                             ^     ^
**          |    |                             |     |
**          |  control codes for text insertion |     |
**          + Entry P value (see MFERR*)        |     |
**                                LEX ID of message     |
**                                              message number
**
**      If parse error (identified by bit3 in R0(S)=1xxx):
**         Address in INBS points to input stream
**         A(A)= address of error within input stream
**
**
** Normal exit conditions from handler if handled (ST, RAM,
** registers, etc.):
**      Carry clear.
**      HEX mode.
**      XM=0.
**      no other requirements -- the message will be suppressed
**
** Normal exit conditions from handler if not handled (ST, RAM,
** registers, etc.):
**      Carry clear.
**      HEX mode.
**      XM=1.
```

```
217             **      R0 can be changed as needed to adjust msg (see MFERR*)
218             **
219             ** Error exit conditions from handler:
220             **      Carry set.
221             **      HEX mode.
222             **      C[0-3] = Error number.
223             **      P= value to select options in MFERR* (caution: do not
224             **         select a parse error in this manner -- A(A) cannot
225             **         pass information back through the poll.  i.e., do
226             **         not set bit3 in P.  If such a thing is necessary,
227             **         the appropriate action is to abort the poll and
228             **         jump directly to BSERR, MFERR or MFERR*.)
229             **
230             ** Available subroutine levels:
231             **      5
232             **
233             ** What registers/RAM may be used if handled?:
234             **      A,B,C,D,D0,D1,P,R0
235             **      (Not available: R1,R2,R3,R4,ST, scratch RAM)
236             **
237             ** What registers/RAM may be used if not handled?:
238             **      A,B,C,D,D0,D1,P,R0 (change R0 only to affect msg)
239             **      (Not available: R1,R3,R4,ST, scratch RAM.
240             **        R2 unavailable except for rare cases when insertion
241             **        text is being passed to the msg routines.)
242             **
243             **      NOTE: If a parse error (bit3 in R0(S)=1xxx),
244             **            then DO NOT call I/OALL, IODALL -- DO
245             **            NOT allocate, deallocate or adjust the
246             **            length of any I/O buffer!  DO NOT change
247             **            the value in AVMEMS or INBS!  (I/O buffer
248             **            routines move I/O buffer memory and change
249             **            AvMemSt.)  These pointers may be changed
250             **            if the error is NOT a parse error.
251             **
252             ** What registers/RAM may be used if error exit (POLL only)?:
253             **      A,B,C,D,D0,D1,P,R0
254             **      (Not available: R1,R2,R4,ST, scratch RAM.
255             **         R3 is unavailable unless the error is a parse error;
256             **         i.e., if R0(S)=1xxx.)
257             **
258             ** NOTE:
259             **      The pERROR poll (and other message polls) are usually
260             **      "handled" without setting XM=0.  This is to allow
261             **      all LEX files to get a chance to intercept the poll.
262             **
263             **      Remember, if a parse error, do NOT change the values
264             **      in AVMEMS or INBS!  This prohibits any adjustment (or
265             **      allocation/deallocation) of I/O buffer length.
266             **
267             **      A LEX file which intercepts the poll has essentially
268             **      four choices:
269             **        1) Abort the error message, continue executing
270             **           or whatever else it wants to do (including
271             **           jumping instead to the warning routine).
```

```
272             **        2) Change the values in R0 to change the format
273             **           of the message:
274             **             i) change R0(4-0) to generate a different
275             **                message
276             **             ii) change R0(5) to select different options
277             **                (see MFERR*).  However, bit3 in R0(5)
278             **                CANNOT be changed!  Bit3 in R0(5) indicates
279             **                a parse error; if you need to change this,
280             **                the appropriate way is to jump directly to
281             **                BSERR, MFERR or MFERR* with your own entry
282             **                conditions.
283             **             iii) change the values in R2 to change text
284             **                insertion (text insertion applies only to
285             **                certain rare messages).
286             **           Then allow the poll to return to the error
287             **           routine with XM=1.
288             **        3) Simply clear XM ("poll handled").  This causes
289             **           the message to be suppressed; message driver
290             **           returns immediately (without setting ERRN or
291             **           ERRL, without checking ON ERROR!)
292             **        4) If error is generated by poll handler, set carry
293             **           and load error number in C(3-0).  This will
294             **           cause the new error to be displayed.
295             **      In addition, the poll handler may perform any
296             **      housekeeping type functions, such as cleaning up
297             **      pending operations.
298             **
299             ** Envisioned application(s):
300             **    A) Foreign Language Translators: if the error message
301             **       number is from the appropriate LEX file, the message
302             **       number in R0 is adjusted to generate the translator's
303             **       message.  (If a type {5} building block is included
304             **       in the message, this will have to be adjusted through
305             **       a nested pTRANS poll, too.  See IDS volume I, chapter
306             **       "Message Handling".)  Set XM=1 and return.
307             **    B) Say a LEX file intercepts all errors, writes the
308             **       message number (ERRN) and line number (ERRL) to a
309             **       file, and suppresses the display of the error.
310             **       When intercepting this poll, it would do the
311             **       necessary processing and return with XM=0.
312             **    C) Say another operating system prevents any error from
313             **       halting execution; instead it issues warnings and
314             **       recovers without user intervention.  It could
315             **       intercept the pERROR poll and jump directly to
316             **       MFWRN so that the error is converted into a warning.
317             **       MFWRN is a subroutine, so processing would return
318             **       to this operating system.
319             **
320             ** History:
321             **
322             **    Date     Programmer            Modification
323             **   --------   ----------   ------------------------------------
324             **  10/05/82   MB           Documentation
325             **  01/27/83   MB           Added "poll handled" suppress
326             **
```

```
327          ********************************************************************
328          ********************************************************************
```

```
329                     EJECT
330             ****************************************************************
331             ****************************************************************
332             **
333             ** Name:(S) pMEM    -  Memory error poll
334             **
335             ** Category:   POLL
336             **
337             ** Type:       FPOLL
338             **
339             ** Purpose:
340             **      Alert LEX files that an "Insufficient Memory" error
341             **      is about to be reported.
342             **
343             ** Should poll be "Handled" (return with XM=0)?:
344             **      Only if you want the message to be entirely suppressed.
345             **      Most applications will "handle" the poll without
346             **      setting XM=0 (see below).
347             **
348             ** Meaning of "Handling" Poll (what does code do if XM=0?):
349             **      It's up to you.  For instance, a LEX file might want to
350             **      intercept all errors and warnings to write them to a
351             **      file; in this case, do your thing and return with XM=0
352             **      so that the message is suppressed.
353             **
354             ** Entry conditions for handler (registers, ST, RAM, etc.):
355             **      B[A] = Poll number.
356             **      HEX mode.
357             **      P=0.
358             **
359             **           F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0
360             **      R0: |  |0  0 |F |                       |     |     |
361             **           ^     ^  ^                        ^     ^
362             **           |     |  |                        |     |
363             **           |     | error flag               |     |
364             **           |   insertion codes              |     |
365             **           + Entry P value (see MFWRN)      |     |
366             **                                 LEX ID of message  |
367             **                                          message number
368             **
369             **
370             ** Normal exit conditions from handler if handled (ST, RAM,
371             ** registers, etc.):
372             **      Carry clear.
373             **      HEX mode.
374             **      XM=0.
375             **      P=0.
376             **      no other requirements -- the message will be suppressed
377             **
378             ** Normal exit conditions from handler if not handled (ST, RAM,
379             ** registers, etc.):
380             **      Carry clear.
381             **      HEX mode.
382             **      XM=1.
383             **      P=0.
```

```
384           **      R0 can be changed as needed to adjust msg (see MEMER*)
385           **
386           ** Available subroutine levels:
387           **      3
388           **
389           ** What registers/RAM may be used if handled?:
390           **      A,B,C,D,D0,D1,P,R0
391           **      (Not available: R1,R2,R3,R4,ST, scratch RAM)
392           **
393           ** What registers/RAM may be used if not handled?:
394           **      A,B,C,D,D0,D1,P
395           **      R0: change R0(3-0) to modify message
396           **      R0: change R0(14-13) to allow text insertion
397           **           (only if you're the LEX file that originated
398           **            the message, and know what you're doing).
399           **      (Not available: R1,R2,R3,R4,ST, scratch RAM.)
400           **
401           **
402           ** NOTE:
403           **      The pMEM poll (and other message polls) are usually
404           **      "handled" without setting XM=0.  This is to allow
405           **      all LEX files to get a chance to intercept the poll.
406           **
407           **      The message number is usually eMEM (18 hex, 24 dec).
408           **      But any LEX file can call the MEMER* routine with its
409           **      own message number; the fact that it called MEMER*
410           **      means that it is reporting insufficient memory.
411           **
412           **      The MEMERR routine uses the leeway area in available
413           **      memory as a building buffer; there is only enough room
414           **      for about 80 characters, plus prefix.  If a poll
415           **      handler substitutes another message number, it cannot
416           **      exceed an 80 character limit (a message should never
417           **      be longer than about 25 characters, anyway).  If it
418           **      does, the computer would enter an infinite MEMERR loop.
419           **
420           **      A LEX file which intercepts the poll has essentially
421           **      four choices:
422           **        1) Abort the error message, continue executing
423           **           or whatever else it wants to do (including
424           **           jumping instead to the warning routine).
425           **        2) Change the values in R0 to change the format
426           **           of the message:
427           **             i) change R0(4-0) to generate a different
428           **                message
429           **            ii) change R0(S) to select different options
430           **                (see MFERR*).  However, bit3 in R0(S)
431           **                CANNOT be changed!  Bit3 in R0(S) indicates
432           **                a parse error; if you need to change this,
433           **                the appropriate way is to jump directly to
434           **                BSERR, MFERR or MFERR* with your own entry
435           **                conditions.
436           **           Then allow the poll to return to the error
437           **           routine with XM=1.
438           **        3) Simply clear XM ("poll handled").  This causes
```

```
439             **          the message to be suppressed; message driver
440             **          returns immediately (without setting ERRN or
441             **          ERRL, without checking ON ERROR!)
442             **       4) Replace the address in level 1 of the RSTK
443             **          (counting from 0) with its own address, so
444             **          that after the message is displayed, processing
445             **          returns to itself.
446             **     In addition, the poll handler can perform any
447             **     housekeeping type functions (such as cleaning up
448             **     pending operations).
449             **
450             **     One other option deserving mention is that of
451             **     generating a memory error which calls for text
452             **     insertion.  For instance, say an external system
453             **     has 6 different files open, and is writing to
454             **     them randomly; it reaches insufficient memory
455             **     while writing to FILE4, so wants to report:
456             **        Write Limit: FILE4
457             **     using a text insertion point to pass "FILE4".
458             **     Before calling MEMER*, set up R2 for insertions.
459             **     When handling the pMEM poll, verify that this is
460             **     indeed your message, adjust R0(14-13) to contain
461             **     the insertion codes, and return with XM=1.
462             **
463             ** Envisioned application(s):
464             **    A) Foreign Language Translators: if the error message
465             **       number is from the appropriate LEX file, the message
466             **       number in R0 is adjusted to generate the translator's
467             **       message.  Set XM=1 and return.
468             **    B) Say a LEX file intercepts all errors, writes the
469             **       message number (ERRN) and line number (ERRL) to a
470             **       file, and suppresses the display of the error.
471             **       When intercepting this poll, it would do the
472             **       necessary processing and return with XM=0.
473             **    C) Say another operating system prevents any error from
474             **       halting execution; instead it issues warnings and
475             **       recovers without user intervention.  It could
476             **       intercept the pERROR poll and jump directly to
477             **       MFWRN so that the error is converted into a warning.
478             **       MFWRN is a subroutine, so processing would return
479             **       to this operating system.
480             **
481             **
482             ** History:
483             **
484             **    Date     Programmer             Modification
485             **  --------   ----------   ----------------------------------
486             **  10/05/82   MB           Documentation
487             **  01/27/83   MB           Added "poll handled" suppress
488             **
489             ****************************************************************
490             ****************************************************************
```

```
491                 EJECT
492               ********************************************************************
493               ********************************************************************
494               **
495               ** Name:(S) MFERR   -  Mainframe BASIC system error
496               **
497               ** Category:   SYSTEM
498               **
499               ** Purpose:
500               **      Generate a BASIC system error from the mainframe
501               **      tables.  See BSERR entry for details.
502               **
503               ********************************************************************
504               ********************************************************************
505               *
506               ********************************************************************
507               ********************************************************************
508               **
509               **
510               ** Name:(S) BSERR   -  BASIC system error
511               **
512               ** Category:   SYSTEM
513               **
514               ** Purpose:
515               **      BSERR -- Generate a BASIC system error.
516               **      MFERR -- First sets C(3-2)=00, then falls into BSERR.
517               **
518               ** Entry: See MFERR*
519               ** Exit:  See MFERR*
520               ** Uses:  See MFERR*. Also S14, S1, S0.
521               ** Calls:     MFERR*
522               **
523               ** Stk lvls:   3
524               **
525               ** NOTE:
526               **      MFERR and BSERR are generally for errors generated by
527               **      the BASIC system, as they exit to the BASIC main loop.
528               **      Those applications which wish to simply display an
529               **      error and return should call MFERR* (a subroutine).
530               **
531               ** Detail:
532               **      MFERR -- Set C(3-2)= 00 for mainframe LEX ID.
533               **      BSERR -- Call MFERR*
534               **               Set NoCont flag (stop execution)
535               **               Clear END statement flag
536               **               Set Error flag
537               **               Exit through BASIC loop
538               **
539               ** History:
540               **
541               **    Date     Programmer            Modification
542               **  --------   ----------   ----------------------------------
543               **  06/29/82   MB           documentation
544               **  03/29/83   JP           Set ERROR flag; Clear END flag
545               **
```

```
546             ********************************************************************
547             ********************************************************************
548             *
549             ********************************************************************
550             ********************************************************************
551             **
552             ** Name:(S) MFERRS  -  Stop BASIC execution for error
553             **
554             ** Category:   SYSTEM
555             **
556             ** Purpose:
557             **      Return to BASIC main loop with status bits set to
558             **      cause execution to stop.
559             **
560             ** Entry:
561             **      P      = 0
562             **
563             ** Exit:
564             **      To ERRRTN (BASIC main loop)
565             **
566             ** Calls:      Exits to ERRRTN (BASIC main loop)
567             **
568             ** Uses.......
569             **  Exclusive: S13, S4, S0
570             **  Inclusive: BASIC main loop uses everything.
571             **
572             ** Stk lvls:   0 (see BASIC main loop: RUNRTN)
573             **
574             ** NOTE:
575             **      Standard entry point to stop BASIC execution because
576             **      of an error.
577             **
578             ** Algorithm:
579             **      Set status NoCont=1
580             **      Set status sENDx=0
581             **      Set status sERROR=1
582             **      Exit to ERRRTN
583             **
584             ** History:
585             **
586             **    Date     Programmer             Modification
587             **  --------   ----------   ----------------------------------
588             **  10/31/83   MB           documented
589             **
590             ********************************************************************
591             ********************************************************************
592 09393 D5  =MFERR  B=C    A              Msg# to B(B).
593 09395 D2           C=0    A              Set LEX#= 00.
594 09397 AE9          C=B    B
595 0939A 7350 =BSERR  GOSUB  MFERR*
596 0939E 85E  =MFERRS ST=1   NoCont         Set "Stop execution" flag.
597 093A1 841          ST=0   sENDx          Clear END statement flag
598 093A4 850          ST=1   sERROR         Set ERROR flag
599 093A7 8C00         GOLONG =ERRRTN        Return to BASIC loop
          00
```

```
600                *
601                *
602                **********************************
603                *
604                *
605 093AD 80FF   MsgPOL CPEX   15              Save entry options in C(S).
606 093B1 04            SETHEX
607 093B3 108           R0=C                   Save msgW, flags in R0.
608 093B6 8C00          GOLONG =POllj          Slow poll.
          00
609                *
```

```
610                 EJECT
611           ****************************************************************
612           ****************************************************************
613           **
614           ** Name:(S) MFWRN   -  Warning/message driver
615           ** Name:(S) MFWRNQ  -  Warning/message driver
616           ** Name:(S) MFWRQ8  -  Warning/message driver
617           **
618           ** Category:   GENUTL
619           **
620           ** Purpose:
621           **      Display warnings and messages from standard message
622           **      tables.
623           **
624           ** Entry:
625           **
626           **  (1)------------------
627           **    |
628           **    |  P= 1xxx   Sound Beep
629           **    |
630           **    |  P= x1xx   Do not store ERRN
631           **    |
632           **    |  P= xx1x   Display message only (Else display
633           **    |              "WRN:" or "WRN L:" prefix, too)
634           **    |
635           **    |  P= xxx1   Display message without setting delay.
636           **    |
637           **
638           **
639           **  (2)------------------
640           **    |
641           **    |  C(3-2)= LEX ID# (hex)  (mainframe ID# = 00)
642           **    |  C(B)= message ID number (hex)
643           **    |
644           **
645           **
646           **  (3)------------------
647           **    |  If desired message has text insertion points:
648           **    |   R2 register: source of text insertion.
649           **    |   C(14):  type of insertion.
650           **    |   C(13):  how many characters in insertion.
651           **    |
652           **    |    R2
653           **    |    -----
654           **    |      = actual output characters if C(14)= 1xxx
655           **    |      = address of output characters if C(14)= 0xxx
656           **    |      = additionally, if C(14)= 0000, upper byte
657           **    |           of R2 contains control nibbles.
658           **    |
659           **    |    C(14)
660           **    |    -----
661           **    |     1xxx   use contents of R2 register as output
662           **    |     0xxx   use address in R2 register to find output
663           **    |
664           **    |     x000   Output is already in ASCII form
```

```
665             **   |
666             **   |         Digit output (digits can be Hex or Dec):
667             **   |      x001   Digit output-- replace leading 0's w/blanks
668             **   |      x010   Digit output-- don't suppress leading 0's
669             **   |      x011   Digit output-- suppress leading 0's
670             **   |
671             **   |
672             **   |         Hex-to-Dec conversions always generate
673             **   |           decimal numbers with 7 digits:
674             **   |      x100   Hex-to-Dec: suppress up to 3 leading 0's
675             **   |      x101   Hex-to-Dec: suppress up to 4 leading 0's
676             **   |      x110   Hex-to-Dec: suppress up to 5 leading 0's
677             **   |      x111   Hex-to-Dec: suppress up to 6 leading 0's
678             **   |
679             **   |
680             **   |     C(13)
681             **   |     -----
682             **   |       For C(14)= 1000 ("ASCII output is in R2")
683             **   |           C(13)= #nibbles-1 to be output. Hence the
684             **   |                  #nibs MUST be even!!; C(13) odd. E.g.,
685             **   |                  if 5 chars for output, C(13)=9.
686             **   |
687             **   |       For C(14)= x0xx (hex or dec digit output)
688             **   |           C(13)= #digits-1 to be output, hence
689             **   |                  no more than 16.
690             **   |
691             **   |       For C(14)= x1xx (hex-to-dec conversion)
692             **   |           C(13)= #digits-1 in number to be converted
693             **   |                  Max hex value for conversion is FFFFF
694             **   |                  (1048575 dec), hence C(13) must be 4
695             **   |                  or less.
696             **   |
697             **   |       For C(14)= 0000 ("ASCII output from DAT1")
698             **   |           C(13)= 0: no output
699             **   |                  1: Send out specified number of
700             **   |                     character; R2(15-14)= #chars-1.
701             **   |                  2: Send out chars until ASCII termin-
702             **   |                     ator is found.  ASCII terminator
703             **   |                     is passed in R2(15-14) (usually
704             **   |                     an FF terminator, but any byte
705             **   |                     value can be used).
706             **   |
707             **
708             **
709             ** Entry for MFWRQ8:
710             **      Same as for MFWRNQ, except that P will be set
711             **      explicitly to 8.  Processing then falls into MFWRNQ.
712             **
713             ** Exit:
714             **      P      = 0
715             **      Carry set
716             **
717             ** Calls:      POLL, SFlag?, KILLKY, fCALC?, CRLFND, UPDCRL,
718             **             SflagC, TBMSID, DOASCI, TBMSTX, A=CUR, AVS=C,
719             **             AVS2DS, CHIRP, XDELAY, CRLFSD, BLDDSP, MFLG=X,
```

```
720          **            R<RST2, RST2<R
721          **
722          ** Uses.......
723          **  Exclusive: A(W), B(W), C(W), D(W), P, D0, D1, R0
724          **             R2 (only if text insertion; otherwise not used)
725          **  Inclusive: Same
726          **
727          ** Stk lvls:   2
728          **
729          ** NOTE:
730          **      If the message constant is eMEM (18 hex), the message
731          **      routines will automatically invoke MEMERR, and issue
732          **      an Insufficient Memory error.
733          **
734          ** Detail:
735          **   Example of text insertion:
736          **      Message #88 in the mainframe is TFM WRN L{5}:{6},
737          **      where {5} indicates an insertion point for a line
738          **      number, and {6} indicates an indirect reference to
739          **      another message.  If we wanted to display
740          **          TFM WRN L145:Syntax      (Syntax is msg #4Bhex)
741          **      we could pass the line number in R2 with the
742          **      appropriate control codes in C (x=don't care):
743          **          R2= xxxxxxx004Bx0145
744          **                            0145= dec digits for output
745          **                      004B= indirect message number
746          **
747          **          C=  xB3xxxxxxxxx0088
748          **                            0088= desired warning message
749          **                 3=#digits-1 to be output
750          **                B=1xxx: use contents of R2
751          **                  x011: digit output, suppress leading 0's
752          **
753          **      Or, alternatively,
754          **          R2= xxxxxxx004Baaaaa
755          **                           aaaaa= address to find digits
756          **                      004B= indirect message number
757          **
758          **          C=  x33xxxxxxxxx0088
759          **                            0088= desired warning message
760          **                 3=#digits-1 to be output
761          **                3=0xxx: use address in R2
762          **                  x011: digit output, suppress leading 0's
763          **
764          **      Or,...
765          **          R2= xxxxxxx004Bxxx91
766          **                              91hex=145 decimal
767          **                      004B= indirect message number
768          **
769          **          C=  xF1xxxxxxxxx0088
770          **                            0088= desired warning message
771          **                 1=#digits-1 to be converted to decimal
772          **                F=1xxx: use contents of B register
773          **                  x111: suppress up to 6 leading 0's
774          **
```

```
775                **      Or,...
776                **          R2= 03xxxxx004Baaaaa
777                **                        aaaaa= address to find ASCII output
778                **                    004B=indirect message number
779                **                03=#characters-1 to be output
780                **
781                **          C=  x01xxxxxxxxx0088
782                **                        0088= desired warning message
783                **                  1= output number of chars found in R2(15-14)
784                **                 0=output is in ASCII form already, resides at
785                **                      address found in R2.
786                **
787                ** History:
788                **
789                **    Date      Programmer              Modification
790                **   --------   ----------   ---------------------------------
791                **  06/29/82    MB           documentation
792                **  01/27/83    MB           Poll error handle, XM=0 suppress
793                **  03/04/83    MB           Saved 3 RSTK levels
794                **  04/11/83    MB           Added KILLKY call.
795                **
796                ********************************************************************
797                ********************************************************************
798 093BC DO     =MFWRN  A=0     A
799 093BE CC             A=A-1   A               A(A)=-1: "Don't check QUIET".
800 093C0 460            GOC     MFWR03          (BET)
801                *
802 093C3 28     =MFWRQ8 P=      8
803 093C5 DO     =MFWRNQ A=0     A               "Check QUIET option".
804 093C7 72EF    MFWR03 GOSUB   MsgPOL          Slow poll with pWARN.
805 093CB 3F             CON(2)  =pWARN
806 093CD 4CC            GOC     BSERR           Error during poll handle.
807                *
808 093D0 8AC            ?A#0    A               Check QUIET?
809 093D3 FO             GOYES   MFWR05          No.
810 093D5 31FF           LC(2)   =flQIET         Yes.
811 093D9 8E00           GOSUBL  =SFlag?         QUIET option set?
          00
812 093DF 400            RTNC                    Yes. Shhh.
813 093E2 8F00    MFWR05 GOSBVL  =KILLKY         Zero KEYDEF (for CALC mode...)
          000
814 093E9 2A             P=      10              To set C(12)=0.
815 093EB 541            GONC    MFER02          (BET) Into error message code.
816                *
817                *
```

```
818                 EJECT
819             ********************************************************************
820             ********************************************************************
821             **
822             ** Name:(S) MFERR*  -  Error message driver
823             ** Name:    MFERR-  -  Error message driver
824             **
825             ** Category:   GENUTL
826             **
827             ** Purpose:
828             **      Display error messages from standard message tables.
829             **
830             ** Entry:
831             **
832             **  (1)-----------------
833             **   |
834             **   |  P=  1xxx    This is a Parse error (i.e., re-
835             **   |                 display input line w/cursor backup)
836             **   |      x1xx    Do not store ERRN
837             **   |                 (Else store ERRN and ERRL)
838             **   |      xx1x    Display msg only (Else display
839             **   |                 "ERR:" or "ERR L:", too)
840             **   |      bit0 not used at present (**)
841             **   |
842             **
843             **
844             **  (2)-----------------
845             **   |
846             **   | C(B)= message ID number in Hex.
847             **   | C(3-2)= LEX ID# in Hex  (=00 for mainframe tbl)
848             **   |
849             **
850             **
851             **  (3)-----------------
852             **   |
853             **   | If P=1xxx  (parse error):
854             **   |    INBS points to first char of INput Buffer, with
855             **   |       a 3 nibble length field preceding it.
856             **   |
857             **   |    D1 points to char in input buffer w/error
858             **   |
859             **   |    A(A)= Address of prompt string for input
860             **   |          re-display (prompt must be enclosed in
861             **   |          delimiters, both sides. Delimiters
862             **   |          can be any byte value. E.g., prompt
863             **   |          string for an editor might look
864             **   |          like  xCmd:x  , where x's are any
865             **   |          matching byte value.)
866             **   |    or =0 For "use BASIC prompt string" (defaults
867             **   |          to the prompt string  3>3, where the
868             **   |          3's are the matching delimiters).
869             **   |
870             **
871             **   (**)  Bit0 of the P register is reserved for future
872             **         applications, as a way for the LEX file which
```

```
873             **          generated the error to communicate with other
874             **          LEX files; this bit can be detected during the
875             **          pERROR poll in R0(S).  The meaning of this bit
876             **          is not yet decided.  In the meantime, bit0 must=0.
877             **
878             ** Entry for MFERR- :
879             **      D0 as C(3-0) above.
880             **
881             **
882             ** Exit:
883             **      P       = 0
884             **
885             ** Calls:       POLL, fCALC?, CRLFND, UPDCRL,
886             **              SflagC, TBMSID, DOASCI, TBMSTX, A=CUR, AVS=C,
887             **              AVS2DS, CHIRP, XDELAY, CRLFSD, BLDDSP, MFLG=X,
888             **              R<RST2, RST2<R.  Might jump to ONERR.
889             **            Parse errors also call:
890             **              CKINF-, DSPBUF, DSPCNA, DSPCHA, CURSFL, CURSRR,
891             **              ESCSEQ
892             **
893             ** Uses.......
894             **  Exclusive: A(W), B(W), C(W), D(W), P, D0, D1, R0
895             **              R2 (only for MFERsp entry with text insertion;
896             **                  otherwise not used)
897             **              S13 is tested for: "Running program?"
898             **                  If you're calling this routine just for
899             **                  message display, watch out for S13!!!
900             **              Available Memory (starting at AvMemSt) is
901             **                  also used as a building buffer for msg.
902             **            PARSE ERRORS also use:
903             **              R3 (stores prompt address and #cursor-rights)
904             **              R1, R2 (used in SENDWD)
905             **              STMTR0 (in CKINF0 and SENDWD)
906             **  Inclusive: Same
907             **
908             ** Stk lvls:    4 (parse errors only)
909             **              2 (all other errors)
910             **
911             ** NOTE:
912             **   Parse errors re-prompt and rebuild the input line.  The
913             **   prompt is built in the display observing WIDTH.  This is
914             **   not a problem with the BASIC prompt (">"), since it is
915             **   only one character; but an external system using a multi-
916             **   character prompt should be aware that the prompt, after
917             **   a parse error, may be split between two lines.  (This
918             **   feature was incorporated to accomodate INPUT prompts.)
919             **
920             **   Messages are built in Available Memory, which is used as
921             **   a temporary buffer.  This can cause a MEMERR; see the
922             **   MEMERR routine for details.
923             **
924             **   If the error message number at entry is the eMEM constant
925             **   (18hex), the message routines will automatically invoke
926             **   the MEMERR routine, and an Insufficient Memory error will
927             **   result.
```

```
928             **
929             **   Any error entering through MFERR* (includes MFERR and
930             **   BSERR) disallows text insertion.  Some applications may
931             **   construct error messages which allow text insertion; if
932             **   you want to issue these messages as errors you have
933             **   three choices:
934             **      1) Issue them without any text insertion (use MFERR*,
935             **         MFERR or BSERR)
936             **      2) Issue them as warnings, made to look like errors
937             **         (use MFWRN) (see IDS volume I, chapter "Message
938             **         Handling").
939             **      3) Call MFERsp entry point (see MFERsp heading).
940             **
941             **
942             ** Detail:
943             **      R0 usage:
944             **        F E D C B A 9 8 7 6 5 4 3 2 1 0
945             **       ---------------------------------
946             **       | |   | |               |       |
947             **       ---------------------------------
948             **        |  |  +- wrng or error    +- msg number
949             **        |  +- insert codes
950             **        +- option flags
951             **
952             ** Algorithm:
953             **     (1) Put option flags in C(S).
954             **         Save options and LEX#, msg# in R0.
955             **         Call POLL
956             **         If Parse error, calculate #backups and store
957             **          with A(A) in R3.
958             **         If eMEM constant, branch to MEMERR.
959             **  MFER.6 If "don't store error#" option go to (2)
960             **          Else, store error# in ERR#.
961             **          If running program (S13=1), store Line#.
962             **     (2) If running program (S13=1), and not a warning,
963             **          and ON ERROR in effect, branch to ONERR.
964             **         If "message text only" option, go to (4)
965             **         Build LEX ID prefix for message.
966             **         Build "ERR" or "ERR L".
967             **         If running program (S13=1), build line#.
968             **         Build ":"
969             **     (4) Build message text.
970             **         Display entire message.
971             **         Beep.
972             **         Send CR, LF.
973             **         If warning, return.
974             **         If not parse error and S13=1, position D0
975             **          to line# or @, return.
976             **         (Parse error:)
977             **           Set up CKINFO for SENDWD, send out prompt.
978             **           Redisplay input line.
979             **           Move cursor far left.
980             **           Send out required # of cursor-rights.
981             **
982             ** History:
```

```
983             **
984             **    Date      Programmer              Modification
985             **  --------    ----------   ----------------------------------
986             **  06/29/82    MB           documentation
987             **  06/18/83    MB           deleted P=xxx1 entry flag
988             **
989             ********************************************************************
990             ********************************************************************
```

```
                EJECT
*****************************************************************
*****************************************************************
**
** Name:(S) MFER42  -  Position D0 to start of BASIC stmt.
**
** Category:   EXCUTL
**
** Purpose:
**      To position D0 to start of BASIC stmt -- to either
**      an "@" character, or the line number.
**
** Entry:
**      PCADDR pointer must be updated already (points
**         to the first token in the BASIC statement).
**      S13=0 if program not running
**         =1 if program running.
**
** Exit:
**      (P unchanged)
**      Carry set: program not running (S13=0 at entry)
**      Carry clear: program running (S13=1 at entry)
**            D0 points to either the "@" character
**              or to the line number at the start of
**              the BASIC statement.
**
** Calls:      D0=PCA, ATCHK
**
** Uses.......
**  Exclusive: D0
**  Inclusive: A(A),D0
**
** Stk lvls:   1
**
** NOTE:
**      This routine does not find the start of a BASIC state-
**      ment -- call CPL#10 for that.  For MFER42, PCADDR must
**      already point to the first token in the statement.
**      This routine simply backs up D0 to the "@" (D0-2),
**      or the line number (D0-6).
**
** Algorithm:
**      If S13=0 (program not running), return.
**      Fetch PC from PCADDR, put in D0.
**      Back D0 up 2 nibbles, to possible "@".
**      ATCHK: If D0 points to "@", rtncc.
**            Else, D0-4 to point to line number.
**
** History:
**
**    Date     Programmer              Modification
**  --------   ----------   ----------------------------------
**  12/08/82   MB           Documentation
**
*****************************************************************
```

```
1046          ************************************************************
```

```
1047                    EJECT
1048            ****************************************************************
1049            ****************************************************************
1050            **
1051            ** Name:(S) MFERsp  -  Error Message With Text Insertion
1052            **
1053            ** Category:   GENUTL
1054            **
1055            ** Purpose:
1056            **      Special entry point into error message handler,
1057            **      allowing text insertion (only in those known messages
1058            **      which have insertion points).
1059            **
1060            ** Entry:
1061            **
1062            **  (1)-----------------
1063            **   |
1064            **   |  R0(S) = entry options as specified for P in MFERR*
1065            **   |
1066            **
1067            **
1068            **  (2)-----------------
1069            **   |
1070            **   | R0(B)= message ID number in Hex.
1071            **   | R0(3-2)= LEX ID# in Hex  (=00 for mainframe tbl)
1072            **   |
1073            **
1074            **
1075            **  (3)-----------------
1076            **   |
1077            **   | Parse errors: Same as condition (3) for MFERR*.
1078            **   |
1079            **
1080            **
1081            **  (4)-----------------
1082            **   |
1083            **   | Text insertion: Same as condition (3) for MFWRN.
1084            **   | (See "Details" under MFWRN for examples.)
1085            **   |
1086            **
1087            **
1088            ** All other details as specified in MFERR* .
1089            **
1090            ** See "NOTE", "Details" and "Algorithm" entries under MFERR*.
1091            **
1092            ** Detail:
1093            **      MFERsp should be called (as a subroutine) as
1094            **      follows:
1095            **
1096            **         <set R2 according to text insertion options>
1097            **         <set C(14-13) according to text insert options>
1098            **         <set C(S) bits according to MFERR* options>
1099            **         <set C(3-0)=message number>
1100            **         R0=C                    Store options, msg_# in R0
1101            **         SETHEX
```

```
1102            **          GOSBVL =POLL            pERROR poll.
1103            **          CON(2) =pERROR
1104            **          CPEX   15               In case poll error, options.
1105            **          P=     12               P value for "error".
1106            **          LCHEX  00F              In case poll error...
1107            **          GOC    LABEL1           CRY=poll error.
1108            **          ?XM=0                   Poll handled?
1109            **          GOYES  LABEL3           Yes! Abort message.
1110            **          C=R0
1111            **          LCHEX  F                C(12)=F for "error" flag.
1112            **  LABEL1  GOSBVL =MFERsp
1113            **  LABEL3  P=     0                (if necessary from ?XM=0
1114            **          .....                      jump, above....)
1115            **
1116            **
1117            ** History:
1118            **
1119            **    Date      Programmer              Modification
1120            **  --------    ----------   ----------------------------------
1121            **  09/22/83    MB           documentation
1122            **
1123            ****************************************************************
1124            ****************************************************************
1125 093EE 136  =MFERR- CDOEX                   LEX ID# and MSG# to C.
1126 093F1 78BF =MFERR* GOSUB  MsgPOL           Slow poll; save entry in R0.
1127 093F5 2F          CON(2) =pERROR
1128            *
1129 093F7 80FF        CPEX   15                In case of POLL error.
1130 093FB 2C          P=     12                To load "00F" in C(14-12).
1131 093FD 4A0         GOC    MFER03            CRY=POLL error. New msg# in C.
1132 09400 118  MFER02 C=R0                     No POLL error. Fetch msg#.
1133 09403 831         ?XM=0                    Poll handled? ("suppress msg"?)
1134 09406 00          RTNYES                   Yes.
1135 09408 32F0 MFER03 LCHEX  00F               C(13-14)= 00: "No insert text".
           0
1136 0940D 108  =MFERsp R0=C
1137 09410 A46         C=C+C  S                 Parse error?
1138 09413 5B2         GONC   MFER05            No.
1139 09416 89A         ?P=    10                Maybe. Warning?
1140 09419 62          GOYES  MFER05            Yes.
1141            *
1142 0941B AF2         C=0    W                 For CSRB below.
1143 0941E 1B6C        DO=(5) =INBS
           6F2
1144 09425 146         C=DAT0 A                 C=address start of parse stream.
1145 09428 133         AD1EX                    A=address of parse error.
1146 0942B EE          C=A-C  A                 C=#nibs to right of start.
1147 0942D 540         GONC   MFER04
1148 09430 D2          C=0    A
1149 09432 81E  MFER04 CSRB                     C=#chars to right of start.
1150 09435 7827        GOSUB  CSLWP9            To C(9-5).
1151 09439 137         CD1EX                    C(A)= prompt address.
1152 0943C 10B         R3=C                     Save both in R3.
1153            *
1154 0943F 110  MFER05 A=R0                     Msg# to A.
```

```
1155 09442 D6          C=A    A          Copy A(4) to C(4).
1156 09444 7650        GOSUB  MEMER1     Load msg# for "Insuff Memory".
1157 09448 8A6         ?A#C   A          MEMERR?
1158 0944B E5          GOYES  MFER.6     No.
1159           *                         YES!! Fall through to MEMERR.
```

```
1160                    EJECT
1161            ********************************************************************
1162            ********************************************************************
1163            **
1164            ** Name:(S) MEMERR  -  Insufficient Memory error
1165            ** Name:(S) MEMERX  -  Insufficient Memory error
1166            **
1167            ** Category:   SYSTEM
1168            **
1169            ** Purpose:
1170            **      Process "Insufficient Memory", exit to BASIC main loop.
1171            **
1172            ** Entry:
1173            **      MEMERR -- No required conditions.
1174            **      MEMERX -- P=entry options as in MEMER*
1175            **
1176            ** Exit:
1177            **      P      = 0
1178            **      Available Memory recoverd (AvMemSt and AvMemEnd
1179            **        collapsed).
1180            **
1181            ** Calls:      MEMER*
1182            **
1183            ** Stk lvls:   3
1184            **
1185            ** NOTE:
1186            **      See MEMER* for all details.
1187            **
1188            ** Detail:
1189            **      MEMERR -- sets P=0
1190            **      MEMERX -- sets C(3-0)= eMEM (18hex)
1191            **                falls into MEMER*
1192            **                exits to BASIC main loop with:
1193            **                      S14=1 (NoCont)
1194            **                      S0=1  (sERROR)
1195            **                      S1=0  (sENDx)
1196            **
1197            ** History:
1198            **
1199            **    Date     Programmer             Modification
1200            **  --------   ----------   ----------------------------------
1201            **  10/05/82   MB           Wrote code, documentation
1202            **
1203            ********************************************************************
1204            ********************************************************************
1205            ********************************************************************
1206            ********************************************************************
1207            **
1208            ** Name:(S) MEMER*  -  Low-level memory error
1209            **
1210            ** Category:   SYSTEM
1211            **
1212            ** Purpose:
1213            **      Display low-level memory error to the user.
1214            **
```

```
1215            ** Entry:
1216            **
1217            **  (1)-----------------
1218            **    |  (same as MFERR*)
1219            **    |
1220            **    |  P= (1xxx)!! Indicates Parse error.  THIS SHOULD
1221            **    |                NEVER BE SET FOR A MEMERR!  MEMER*
1222            **    |                collapses AvMemSt, causing the
1223            **    |                input buffer (address in INBS) to
1224            **    |                be destroyed!
1225            **    |       x1xx    Do not store ERRN
1226            **    |                (Else store ERRN and ERRL)
1227            **    |       xx1x    Display msg only (Else display
1228            **    |                "ERR:" or "ERR L:", too)
1229            **    |       bit0 not used at present (**)
1230            **    |
1231            **
1232            **
1233            **  (2)-----------------
1234            **    | (same as MFERR*)
1235            **    |
1236            **    | C(B)= message ID number in Hex.
1237            **    | C(3-2)= LEX ID# in Hex  (=00 for mainframe tbl)
1238            **    |
1239            **
1240            **
1241            **  (3)-----------------
1242            **    | (same as MFERR*)
1243            **    |
1244            **    | NEVER CALL MEMER* AS A PARSE ERROR! (I.e., never
1245            **    | enter with P=1xxx.)
1246            **    |
1247            **
1248            **   (**)  Bit0 of the P register is reserved for future
1249            **         applications, as a way for the LEX file which
1250            **         generated the error to communicate with other
1251            **         LEX files; this bit can be detected during the
1252            **         pMEM poll in R0(S).  The meaning of this bit is
1253            **         not yet decided.  In the meantime, bit0 must=0.
1254            **
1255            ** Exit:
1256            **       P       =  0
1257            **
1258            ** Calls:       FPOLL, COLLAP, CLCOLL, AUTCLR, TRNFCK,
1259            **              MFER.6 (MFER.6 is an entry point in MFERR* --
1260            **                see MFERR* for more details)
1261            **
1262            ** Uses.......
1263            **  Exclusive: A(W), B(W), C(W), D(W), P, D0, D1, R0
1264            **             S13 is tested for: "Running program?"
1265            **                 If you're calling this routine just for
1266            **                 message display, watch out for S13!!!
1267            **             Available Memory (starting at AvMemSt) is
1268            **                 also used as a building buffer for msg.
1269            **   Inclusive: Same
```

```
1270            **
1271            ** Stk lvls:   2
1272            **
1273            ** NOTE:
1274            **   The entry point MEMER* allows ANY message to be
1275            **   reported in lieu of "Insufficient Memory", and still
1276            **   be handled as a memory error.  This means you can
1277            **   display, say, "Out of Scratch Area" as a way of
1278            **   reporting a memory error.  This capability is included
1279            **   to allow external systems to generate memory errors
1280            **   and report them as they desire.  But this capability
1281            **   can cause serious conditions (such as an infinite
1282            **   MEMERR loop) if some rules are not followed:
1283            **       1) Never invoke MEMER* (or MEMERR or MEMERX) as
1284            **          a parse error.
1285            **       2) Any error entering through MEMER* (includes
1286            **          MEMERR and MEMERX) disallows text insertion.
1287            **          This can be overridden in the pMEM poll.  But
1288            **          never use a message which contains a type{5}
1289            **          insertion!!!  A type{5} insertion may cause
1290            **          a slow pTRANS poll to be issued, which may
1291            **          cause an infinite MEMERR loop.
1292            **
1293            **   The preferred way for a LEX file operating in the
1294            **   BASIC system to generate a different memory error
1295            **   (i.e., other than "Insufficient Memory"), is to call
1296            **   MEMERR and then intercept the pMEM poll to change the
1297            **   message number or options.  On the other hand, a
1298            **   LEX file which wants to generate a memory error which
1299            **   takes text insertions should set up the insertion
1300            **   codes in R2, call MEMER* with the appropriate message
1301            **   number, and adjust C(14-13) during the pMEM poll.
1302            **
1303            ** Detail:
1304            **       R0 usage:
1305            **         F E D C B A 9 8 7 6 5 4 3 2 1 0
1306            **        ---------------------------------
1307            **        | |   |F|               |       |
1308            **        ---------------------------------
1309            **         |  |  +- error code       +- msg number
1310            **         |  +- insert codes
1311            **         +- option flags
1312            **
1313            ** Algorithm:
1314            **      (1) Put option flags in C(S).
1315            **          Save options and LEX#, msg# in R0.
1316            **          Set C(14-12)=00F (suppresses text insertions)
1317            **          Call FPOLL
1318            **          Collapse Available Memory
1319            **          Turn off AUTO mode
1320            **          Check if TRANSFORM in effect (this essentially
1321            **            include TRANSFORM in the poll); if so
1322            **            branch back to TRANSFORM.
1323            **          Jump to MFER.6  (see MFERR*)
1324            **
```

```
1325            ** History:
1326            **
1327            **    Date     Programmer              Modification
1328            **  --------   ----------   ----------------------------------
1329            **  10/05/82   MB           documentation
1330            **
1331            ********************************************************************
1332            ********************************************************************
1333            *
1334 0944D 20    =MEMERR P=     0
1335 0944F 7B40  =MEMERX GOSUB  MEMER1          "Insufficient Memory".
1336 09453 7400          GOSUB  MEMER*          Subroutine to allow external
1337            *                                 call to MEMER*
1338 09457 664F          GOTO   MFERRS          (goes to RUNRTN)
1339            *
1340            *
1341 0945B 04    =MEMER* SETHEX
1342 0945D 80FF          CPEX   15              Save P in C(S).
1343 09461 2C            P=     12              C(12)=F identifies error;
1344 09463 32F0          LCHEX  00F               C(14-13): no insertions;
           0
1345 09468 108           RO=C
1346 0946B 8E00          GOSUBL =FPoll          Fast poll. (Sets P=0.)
           00
1347 09471 1F            CON(2) =pMEM
1348            *
1349 09473 831           ?XM=0                  Poll handled?
1350 09476 00            RTNYES                 Yes. Suppress message.
1351 09478 7000          GOSUB  =COLLAP         Collapse AVMEME to FORSTK.
1352 0947C 8F00          GOSBVL =CLCOLL         Collapse AVMEMS to CLCSTK.
           000
1353 09483 8E00          GOSUBL =AUTCLR         Turn off AUTO mode.
           00
1354            *
1355 09489 8E00          GOSUBL =TRNFCK         TRANSFORM in effect?
           00
1356 0948F 491           GOC    MFER.6          No.
1357            *
1358 09492 06            RSTK=C                 Yes. Control addr to stack.
1359 09494 AF2           C=0    W
1360 09497 15D5          DAT1=C 6               Clear addr and trnsfm flag.
1361 0949B 845           ST=0   5               Clear TRANSFORM's flag...
1362 0949E 1A00  MEMER1  D0=(4) =eMEM           Load eMEM as error number
           00
1363 094A4 13E           CDOXS                    (short for MFER05 call).
1364 094A7 02            RTNSC                  CRY set: error.
1365            *
1366            ****************************************************
1367            *
1368 094A9 8F00  MFER.6  GOSBVL =R<RST2         Save 3 levels of RSTK.
           000
1369            *----------------------------------------------------
1370            *
1371 094B0 8E00          GOSUBL =fCALC?         In CALC mode?
           00
```

```
1372 094B6 580          GONC   MFER+6          No.
1373 094B9 8E00         GOSUBL =CRLFND         Yes. CR-LF to clear display.
           00
1374 094BF 7812  MFER+6 GOSUB  ESC"<"          Turn cursor off.
1375 094C3 86D          ?ST=0  13              Running program?
1376 094C6 60           GOYES  MFER07          No.
1377 094C8 7B12         GOSUB  UPDCRL          Update Current Line @ PCADDR
1378             *
1379             * Now hash entry options:  at entry, P= abcd
1380             *
1381             *                        C(S)     C(12)
1382             *                        ====     =====
1383             *              Error:    bcd0     10a0
1384             *              Warning:  bcx0     a1xd      (x= don't care)
1385             *                        ~~~~     ~~~~
1386             *      don't store ERRN -+|||     |||+- inhibit delay
1387             *         msg text only --+||     ||+-- parse error
1388             *         skip ON ERROR ---+|     |+--- warning
1389             *            (not used) ----+     +---- beep
1390             *
1391             *
1392 094CC 118   MFER07 C=R0                      Codes and msg# to C.
1393 094CF 80DF         P=C    15
1394 094D3 7E61         GOSUB  STOSW3          Options to STO.
1395 094D7 852          ST=1   warn            Set "Warning" flag.
1396 094DA 890          ?P=    0               Warning or system message?
1397 094DD B0           GOYES  MFER08          Yes.
1398 094DF 20           P=     0               No, error.
1399 094E1 0B           CSTEX                  Status bits back to C.
1400 094E3 308          LCHEX  8               STbeep =1, others=0.
1401 094E6 0B           CSTEX
1402 094E8 A46   MFER08 C=C+C  S               Parse?
1403 094EB 550          GONC   MFER09          No.
1404 094EE 851          ST=1   parse           Yes. (No matter for warning.)
1405             *
1406 094F1 A46   MFER09 C=C+C  S               Store ERRN#?
1407 094F4 4F1          GOC    MFER11          No.
1408 094F7 1A4E         DO=(4) =ERR#           Yes.
           7F
1409 094FD 15C3         DATO=C 4               Write LEX#, msg#.
1410 09501 86D          ?ST=0  13              Running program?
1411 09504 01           GOYES  MFER11          No. Don't store ERRL.
1412 09506 163          DO=DO+ 4               Yes. DO=CURRL.
1413 09509 15A3         A=DATO 4               Take current line#,
1414 0950D 163          DO=DO+ 4                  and write
1415 09510 1583         DATO=A 4                  it to ERRL.
1416             *
1417 09514 872   MFER11 ?ST=1  warn            Warning?
1418 09517 03           GOYES  MFER15          Yes. Skip ON ERROR check.
1419             *                             Error: first clear flNOFN.
1420 09519 316D         LC(2)  =flNOFN         Clear "No user-defnd fcn allowed".
1421 0951D 8E00         GOSUBL =SflagC         NOTE: flNOFN is chosen such that
           00
1422             *                             the SFLAGC routine does not use
1423             *                             C(S). I.e., the hex value of
```

```
1424            *                               flNOFN cannot have 0 or 8 in the
1425            *                               least significant nibble.
1426            *
1427 09523 86D          ?ST=0  13               Running program?
1428 09526 12           GOYES  MFER15            No. Don't bother check ON ERROR.
1429 09528 1A38         DO=(4) =ERRSUB          Yes. If ON ERROR GOSUB in effect
           6F
1430 0952E 146          C=DAT0 A                  execution stops with error.
1431 09531 8AE          ?C#0   A
1432 09534 31           GOYES  MFER15           ON ERROR GOSUB in effect.
1433 09536 164          DO=DO+ (ERRADR)-(ERRSUB)   If ON ERROR address
1434 09539 146          C=DAT0 A                     is not zero,
1435 0953C 8AA          ?C=0   A                     jump to it.
1436 0953F 80           GOYES  MFER15
1437 09541 8C00         GOLONG =ONERR           (Does a P=0.)
           00
1438            *
1439 09547 71E5  MFER15 GOSUB  DO=AVS           Set DO= AvMemSt for build buffer.
1440 0954B A46          C=C+C  S                Text only?
1441 0954E 4B5          GOC    MFER33           Yes.
1442 09551 118          C=R0                    LEX#, msg# to C.
1443 09554 D5           B=C    A                To B.
1444 09556 7854         GOSUB  TBMSID           LEX ASCII ID to buffer.
1445 0955A 1F00         D1=(5) =ERRMST          Err msg prefix address.
           000
1446 09561 862          ?ST=0  warn             Error?
1447 09564 80           GOYES  MFER19           Yes. Use error message prefix.
1448 09566 1E00         D1=(4) (=WRNMST)+13     No. To warning msg prefix.
           00
1449 0956C 304   MFER19 LCHEX  4                Codes for DOASCI: send out 5
1450 0956F 73B2         GOSUB  DOASC*             chars from DAT1.
1451 09573 183          DO=DO- 4                Back up 2 chars (past " L").
1452 09576 86D          ?ST=0  13               Running program?
1453 09579 92           GOYES  MFER27           No. Back up over " L" in prefix.
1454 0957B 1F8E         D1=(5) =CURRL           Yes. Leave " L" in prefix.
           7F2
1455 09582 147          C=DAT1 A                Read in 4-digit line# (ERRL#).
1456 09585 F2           CSL    A                Zero out nib 0.
1457 09587 8AE          ?C#0   A                Zero line#?
1458 0958A D0           GOYES  MFER25           No. (Current BASIC file).
1459 0958C 31E7         LCASC  \~\              Yes. (Binary, PASCAL,...)
1460 09590 7883         GOSUB  DOASu+           Put out "~".
1461 09594 511          GONC   MFER29           (BET)  Put out another "~".
1462            *
1463 09597 163   MFER25 DO=DO+ 4                Back out past " L".
1464 0959A 3133         LCHEX  33               Dec# output.
1465 0959E 7192         GOSUB  DOASCI           Send 4 digit line# to buffer.
1466 095A2 31A3  MFER27 LCASC  \:\              End message prefix
1467 095A6 7273  MFER29 GOSUB  DOASu+             with ":".
1468            *
1469 095AA 29    MFER33 P=     9                To zero out R0(9-0).
1470 095AC 118          C=R0                    Message number to C.
1471 095AF 7AF3         GOSUB  TBMSTX           Complete msg to build buffer.
1472            *------------------  Note: MEMERR from TBMSTX will destroy
1473            *------------------  original status bits S0-S3 (in R0(12)).
```

```
1474 095B3 100          RO=A                Save old AvMemSt addr in RO.
1475 095B6 E0           A=A-B  A            A=-length of msg in nibs.
1476 095B8 310C         LC(2)  96*2         Check if msg exceeds 96 char bfr.
1477 095BC C2           C=A+C  A            Maximum start posn for msg
1478              *                           in order not to exceed buffer.
1479 095BE 8E00         GOSUBL =A=CUR       Set A(B)=cursor posn (#bytes)
          00
1480 095C4 C4           A=A+A  A            Cursor posn (#nibs).
1481 095C6 9EA          ?A<=C  B            Present posn OK?
1482 095C9 B1           GOYES  MFER35       Yes.
1483              *                         No. Msg won't fit; send CR-LF.
1484 095CB D9           C=B    A
1485 095CD 8F00         GOSBVL =AVS=C       Set AvMemSt=end msg, to protect.
          000
1486 095D4 8E00         GOSUBL =CRLFND      Send CR-LF.
          00
1487 095DA 118          C=RO                Address original AvMemSt.
1488 095DD 8F00         GOSBVL =AVS=C       Reset AvMemSt.
          000
1489 095E4 7021  MFER35 GOSUB  AVS2DS       Display prefix and message.
1490 095E8 863          ?ST=0  beep         Beep?
1491 095EB 80           GOYES  MFER37       No.
1492 095ED 8E00         GOSUBL =CHIRP       Yes.
          00
1493 095F3 860   MFER37 ?ST=0  delay        Inhibit delay?
1494 095F6 80           GOYES  MFER39       No.
1495 095F8 8E00         GOSUBL =XDELAY      Yes.
          00
1496 095FE 8E00  MFER39 GOSUBL =CRLFSD      CR, LF, set delay.
          00
1497 09604 8E00         GOSUBL =BLDDSP      Needed for Inhibit delay.
          00
1498 0960A 30F          LCHEX  F
1499 0960D 8F00         GOSBVL =MFLG=X      Set MLFFLG=F to restore CKINFO.
          000
1500              *
1501 09614 8F00         GOSBVL =RST2<R      Restore 3 levels of RSTK.
          000
1502              *
1503 0961B 872          ?ST=1  warn         Warning message?
1504 0961E 02           GOYES  STOSWP       Yes.
1505 09620 871          ?ST=1  parse        Parse error?
1506 09623 20           GOYES  MFER41       Yes. Just sets carry.
1507 09625 7510  MFER41 GOSUB  STOSWP       Swap back original status.
1508 09629 4C2          GOC    MFER51       Parse error.
1509 0962C 86D  =MFER42 ?ST=0  13           Running program?
1510 0962F 00           RTNYES              No.
1511              *
1512              *****  CLEAN-UP: FOR EXECUTION ERRORS ONLY  *******
1513              *
1514 09631 7205         GOSUB  DO=PCA
1515 09635 181          DO=DO- 2
1516 09638 8C00         GOLONG =ATCHK       DO to @ or line#.
          00
1517              *
```

```
1518            ************************************************************
1519            *
1520 0963E 118   STOSWP C=R0                  Fetch status bits from R0.
1521 09641 80FC  STOSW1 CPEX   12             Swap
1522 09645 0B    STOSW3 CSTEX                     STO
1523 09647 80F0         CPEX   0                    and
1524 0964B 0B           CSTEX                          C(12),
1525 0964D 80FC         CPEX   12
1526 09651 108          R0=C                  put back in R0.
1527 09654 01           RTN                   Preserve carry!
1528            *
```

```
1529                    EJECT
1530            *
1531            ********************************************************************
1532            ********************************************************************
1533            **
1534            ** Name:(S) DONNA   -  Re-prompt input line
1535            **
1536            ** Category:   DSPUTL
1537            **
1538            ** Purpose:
1539            **      Re-display an input line, with a prompt, and position
1540            **      the cursor to any desired point in the line.
1541            **
1542            ** Entry:
1543            **      R3(A)= Address of prompt. The prompt can be any
1544            **             ASCII string, delimited with two matching
1545            **             bytes (delimiters can be any byte value).
1546            **      R3(9-5)= Number of cursor-rights to position the
1547            **             cursor within the input stream (counted
1548            **             from the first input character).
1549            **      INBS contains the address of the input buffer;
1550            **             the length of the input buffer is contained
1551            **             in the three nibbles preceding the buffer.
1552            **
1553            ** Exit:
1554            **      P      = 0
1555            **      Carry set.
1556            **      D1=FFFFF.
1557            **
1558            ** Calls:      CKINF-, DSPBUF (SENDWD), ESCSEQ, DSPCNA, DSPCHA,
1559            **             CURSFL, CURSRR.
1560            **
1561            ** Uses.......
1562            **  Exclusive: A(W),B(W),C(W),D(W),D0,D1,P
1563            **  Inclusive: same plus R1,R2 (in SENDWD), STMTRO (in CKINF-)
1564            **
1565            ** Stk lvls:   4
1566            **
1567            ** NOTE:
1568            **      The prompt is built in the display observing WIDTH;
1569            **      the input line is displayed without observing WIDTH.
1570            **      Any single-character prompt will not have to worry
1571            **      about this, but a multi-character prompt may be
1572            **      split between two lines if WIDTH is short.
1573            **
1574            **      The length of the input buffer (found in the three
1575            **      nibbles preceding the buffer) must be one greater
1576            **      than the number of characters (usually this length
1577            **      includes a 0D terminator at the end of a BASIC
1578            **      input line).
1579            **
1580            **      Example of prompt:
1581            **        Say an editor uses the prompt "Cmd:". The address
1582            **        in R3(A) would point to the characters  xCmd:x
1583            **        where the x's are delimiters, any matching byte
```

```
1584            **        value.
1585            **
1586            ** Algorithm:
1587            **      Turn off cursor.
1588            **      Set up CKINFO.
1589            **      Display prompt.
1590            **      Redisplay input line.
1591            **      Send out a null character (in case input line had
1592            **        zero length, this clears display buffer)
1593            **      Cursor far left.
1594            **      Count cursor-rights, using count in R3(9-5).
1595            **
1596            ** History:
1597            **
1598            **    Date     Programmer             Modification
1599            **  --------   ----------   ----------------------------------
1600            **  10/05/82   MB           Documentation
1601            **
1602            ********************************************************************
1603            ********************************************************************
1604            *
1605            ********************************************************************
1606            ********************************************************************
1607            **
1608            ** Name:(S) CURSRT  -  Count cursor-rights
1609            **
1610            ** Category:   DSPUTL
1611            **
1612            ** Purpose:
1613            **      Send out a cursor-far left, then send out a given
1614            **      number of cursor-rights.
1615            **
1616            ** Entry:
1617            **      C(A)= count of cursor-rights.
1618            **
1619            ** Exit:
1620            **      P      =  0
1621            **      Carry set.
1622            **      D1=FFFFF.
1623            **
1624            ** Calls:      CURSFL, CURSRR
1625            **
1626            ** Uses.......
1627            **  Exclusive: D1
1628            **  Inclusive: A(W),B(W),C(W),D(W),D0,D1,P
1629            **
1630            ** Stk lvls:   3
1631            **
1632            ** Algorithm:
1633            **      Copy counter to D1
1634            **      Cursor far-left
1635            **      Count cursor-rights until D1 carries
1636            **
1637            ** History:
1638            **
```

```
1639            **    Date     Programmer            Modification
1640            **  --------   ----------   ---------------------------------
1641            **  10/05/82   MB           documentation
1642            **
1643            ************************************************************
1644            ************************************************************
1645 09656      MFER51
1646 09656      =DONNA
1647 09656 7180         GOSUB  ESC"<"          Cursor off (nonreadable prompt).
1648 0965A 1F48         D1=(5) =PRP3>3         Point to BASIC prompt ">".
           690
1649 09661 11B          C=R3
1650 09664 8AA          ?C=0   A               Default BASIC prompt string?
1651 09667 11           GOYES  MFER53          Yes. D1= BASIC prompt already.
1652 09669 8F00         GOSBVL =CKINF-         No. Set up output info for
           000
1653 09670 11B          C=R3                     display.
1654 09673 135          D1=C                   Use caller's prompt string.
1655 09676 24           P=     2^((width?)-(InhEOL))  Flag for "observe width".
1656 09678 D1    MFER53 B=0    A               Clear counter.
1657 0967A 14B          A=DAT1 B               Read delimiter to match.
1658 0967D 171          D1=D1+ 2               To first char.
1659 09680 7F90         GOSUB  DSPBUF          Display till delimiter match.
1660 09684 33E3 =PRP3>3 LCASC  \3>\            Cursor on. Reads "NIBASC \3>3\" !
           33
1661 0968A 7350         GOSUB  ESCSEq
1662 0968E 1B6C         D0=(5) =INBS
           6F2
1663 09695 75A4         GOSUB  D0=DT0          Set A and D0= input buffer addr.
1664 09699 131          D1=A                   Also D1.
1665 0969C 182          D0=D0- 3               To length field.
1666 0969F AF0          A=0    W               For ASRB below.
1667 096A2 15A2         A=DAT0 3               Read length.
1668 096A6 81C          ASRB                   Length in bytes.
1669 096A9 CC           A=A-1  A               Less one for 0D terminator.
1670 096AB 460          GOC    MFER55          In case 0 output chars, ...
1671 096AE 7F60         GOSUB  DSPCNA          Output chars by count.
1672 096B2 D0    MFER55 A=0    A               Force a null char (needed
1673 096B4 8E00         GOSUBL =DSPCHA           for INPUT"","";X )
           00
1674 096BA 11B          C=R3                   #cursor-rights to C(9-5).
1675 096BD 7B84         GOSUB  CSRWP9          C(A)=#cursor-rights.
1676 096C1 135  =CURSRT D1=C                   D1 counts #cursor-rights.
1677 096C4 8F00         GOSBVL =CURSFL         Cursor to start of line.
           000
1678            *
1679 096CB 1C0   MFER57 D1=D1- 1               Count #cursor-rights.
1680 096CE 400          RTNC
1681 096D1 8F00         GOSBVL =CURSRR         Send Cursor-Right.
           000
1682 096D8 52F          GONC   MFER57          (BET)
1683            *
1684            ************************************************************
1685            *
1686 096DB 20    ESC"<" P=     0               "ESC<" = cursor off.
```

```
1687 096DD 31C3          LCASC  \<\
1688 096E1 8C00  ESCSEq GOLONG =ESCSEQ        Escape sequence.
           00
1689             *
```

```
1690                    STITLE Update Current Line
1691                ********************************************************************
1692                ********************************************************************
1693                **
1694                ** Name:    UPDCRL  -  Update Current Line @ PCADDR
1695                **
1696                ** Category:   EXCUTL
1697                **
1698                ** Purpose:
1699                **      Compute Line # @ PCADDR and update CURRL
1700                **
1701                ** Entry:
1702                **      P      =  0
1703                **      PCADDR @  Statement Length of statement to update
1704                **
1705                ** Exit:
1706                **      P      =  0
1707                **      Carry Clear
1708                **
1709                **      If current file not BASIC or PCADDR does not point
1710                **        into current file, sets CURRL=0.
1711                **
1712                **
1713                ** Calls:      DO=PCA, CPL#15
1714                **
1715                ** Uses.......
1716                **  Exclusive: C(A)
1717                **  Inclusive: A(A)-D(A),D0,D1
1718                **
1719                ** Stk lvls:   3
1720                **
1721                ** History:
1722                **
1723                **    Date     Programmer             Modification
1724                **  --------   ----------   ----------------------------------
1725                **  03/07/83   JP           Added routine
1726                **  04/11/83   MB           Added BSTYCK call.
1727                **  04/13/83   JP           CURRL=0 if PCADDR not in prog scope
1728                **  04/18/83   JP           Removed SCOPCK call
1729                **  04/18/83   JP           CPL#10 replaces COMPL# call
1730                **
1731                ********************************************************************
1732                ********************************************************************
1733 096E7 8E00  =UPDCRL GOSUBL =BSTYCK        BASIC type check.
           00
1734 096ED 521          GONC   UPDCR1         NC= not BASIC. Set CURRL=0.
1735 096F0 7344         GOSUB  DO=PCA         Set DO= PC address.
1736 096F4 8E00         GOSUBL =CPL#10        D1= line# of curr line.
           00
1737 096FA 147          C=DAT1 A              C=line#.
1738 096FD 540          GONC   UPDCR3         NC= PCADDR not in current file.
1739 09700 D2    UPDCR1 C=0    A              If not current BASIC file, set=0.
1740 09702           =savel0
1741 09702 8C00  UPDCR3 GOLONG =SAVEL0        Save in CURRL.
           00
```

```
1742                    STITLE Display Utilities
1743            ********************************************************************
1744            ********************************************************************
1745            **
1746            ** Name:(S) AVS2DS  -  AvMemSt to display
1747            **
1748            ** Category:   DSPUTL
1749            **
1750            ** Purpose:
1751            **      Send ASCII stored at AvMemSt to display.
1752            **
1753            ** Entry:
1754            **      P = 0   (P is used to select options, must =0!)
1755            **      ASCII characters reside in memory starting at
1756            **         AvMemSt; an FF byte must immediately follow
1757            **         the characters.
1758            **
1759            ** Exit:
1760            **      P       = 0
1761            **      Carry clear.
1762            **
1763            ** Calls:      DO=AVS, DSPBUF
1764            **
1765            ** For all other details, see DSPBUF.
1766            **
1767            **
1768            ** History:
1769            **
1770            **    Date     Programmer             Modification
1771            **  --------   ----------   ----------------------------------
1772            **  06/25/82   MB           documentation
1773            **
1774            ********************************************************************
1775            ********************************************************************
1776            *
1777            ********************************************************************
1778            ********************************************************************
1779            **
1780            ** Name:(S) DSPCNA  -  Display by count
1781            ** Name:(S) DSPCNB  -  Display by count
1782            ** Name:(S) DSPCNO  -  Display by count
1783            **
1784            ** Category:   DSPUTL
1785            **
1786            ** Purpose:
1787            **      Send ASCII characters to display, by count.
1788            **      DSPCNO -- Counter in B(A), use Output Buffer.
1789            **      DSPCNB -- Counter in B(A), use DAT0.
1790            **      DSPCNA -- Counter in A(A), use DAT0.
1791            **
1792            ** Entry:
1793            **      DSPCNO -- #characters-1 in B(A), output resides in
1794            **                Output Buffer (address in OUTBS).
1795            **      DSPCNB -- #characters-1 in B(A), D0 points to output
1796            **      DSPCNA -- #characters-1 in A(A), D0 points to output
```

```
1797            **
1798            ** Exit:
1799            **      P       = 0
1800            **      Carry clear.
1801            **
1802            ** Calls:       DOOUTBS (DSPCNO only), DSPBUF
1803            **
1804            ** For all other details, see DSPBUF
1805            **
1806            ** History:
1807            **
1808            **    Date      Programmer              Modification
1809            **  --------    ----------   ------------------------------------
1810            **  06/25/82    MB           documentation
1811            **
1812            ********************************************************************
1813            ********************************************************************
1814            *
1815            ********************************************************************
1816            ********************************************************************
1817            **
1818            ** Name:(S) DSPBUF  -  Send a buffer of chars to display
1819            **
1820            ** Category:   DSPUTL
1821            **
1822            ** Purpose:
1823            **      Send a buffer of characters to display, allowing
1824            **        1) terminate buffer on count or FF byte.
1825            **        2) observe WIDTH or not.
1826            **
1827            ** Entry:
1828            **
1829            **   (1)--------------------
1830            **     |
1831            **     | P=  0    Send out characters until a terminator
1832            **     |          byte is encountered (terminator byte is
1833            **     |          passed in A(B)).  Do not observe WIDTH
1834            **     |          (i.e., do not split up display into
1835            **     |          WIDTH-sized chunks).
1836            **     |
1837            **     |     2    Count characters.  Send out characters
1838            **     |          until counter decrements (counter passed
1839            **     |          in A(A)).  Do not observe WIDTH.
1840            **     |
1841            **     |     4    Send out characters until a terminator
1842            **     |          byte is encountered (terminator byte is
1843            **     |          passed in A(B)).  Observe WIDTH.
1844            **     |
1845            **     |   Note: The combination "Count characters and
1846            **     |         observe WIDTH" is performed by SENDWD.
1847            **     |
1848            **
1849            **
1850            **   (2)--------------------
1851            **     |
```

```
1852            **     |  If P=2 (send by count):
1853            **     |    A(A)= #characters in buffer.
1854            **     |
1855            **     |  If P=0 or 4 (send until terminator):
1856            **     |    A(B)= terminator byte
1857            **     |    B(A)= 0  (used for separate counter).
1858            **     |
1859            **
1860            **
1861            **    (3)--------------------
1862            **     |
1863            **     |  D0 points to output buffer.
1864            **     |
1865            **
1866            ** Exit:
1867            **       P      =  0
1868            **       Carry clear.
1869            **
1870            ** Calls:       CSLWP9
1871            **              DSPCHA (for entry P=2 only)
1872            **              SENDWD (for entry P=0 or 4 only)
1873            **
1874            ** Uses.......
1875            **  Exclusive: P,A,C,D1,R0(10-5)
1876            **  Inclusive: B,D,D0
1877            **              R1,R2 and STMTR0 (in SENDWD) for P=0 or 4 only
1878            **
1879            ** Stk lvls:   3
1880            **
1881            ** NOTE:
1882            **       R0(15-11) and R0(4-0) are not touched by this routine.
1883            **
1884            ** Algorithm:
1885            **       Swap P (options) into ST1, swap ST1 into R0(10).
1886            **    1) If by count, decrement counter; if carry, goto 2).
1887            **       If by terminator, test match; if match, goto 2).
1888            **       If observe width, count buffer length in B(A),
1889            **         go to 1).
1890            **       Save counter or match in R0.
1891            **       Send out character (DSPCHA).
1892            **       Fetch counter or match in R0.
1893            **       Go to 1).
1894            **    2) If observe width, call SENDWD with length in A.
1895            **       Restore ST1 from R0(10).
1896            **
1897            ** History:
1898            **
1899            **     Date      Programmer              Modification
1900            **   --------    ----------   -----------------------------------
1901            **   06/25/82    MB           documentation
1902            **
1903            ***************************************************************
1904            ***************************************************************
1905            *
1906 09708 7024 =AVS2DS GOSUB  D0=AVS         AvMemSt buffer to display.
```

```
1907 0970C 131          D1=A                    D1=AvMemSt.
1908 0970F D0           A=0     A               Use FF terminator:
1909 09711 CC           A=A-1   A                 A(B)= FF.
1910 09713 4F0          GOC     DSPBUF          (BET)
1911                *
1912                ****************************************************************
1913                *
1914 09716 8E00 =DSPCN0 GOSUBL  =DOOUTB         C= <OUTBS>.
           00
1915 0971C 135          D1=C                    Set D1= <OUTBS>.
1916 0971F D4   =DSPCNB A=B     A               #chars in buffer to A.
1917 09721 22   =DSPCNA P=      2^((cntr?)-(InhEOL))    Flag for "count chars".
1918                *                           Fall into DSPBUF.
1919                *
1920 09723 0B   =DSPBUF CSTEX                   Replace S7-S4 with
1921 09725 80F1         CPEX    1                  pointer value;
1922 09729 0B           CSTEX                      S7-S4 to pointer.
1923 0972B 118          C=R0                    Save R0(15-11).
1924 0972E 80FA         CPEX    10              Save S7-S4 in R0(10).
1925 09732 7B24         GOSUB   CSLWP9          R0(A) to R-(9-5).
1926 09736 D6           C=A     A               Counter/match to C.
1927                *
1928 09738 14B   DSPBF1 A=DAT1  B               Fetch char.
1929 0973B 875          ?ST=1   cntr?           Counter?
1930 0973E A2           GOYES   DSPBF9          Yes.
1931 09740 962          ?A=C    B               No. Match delimiter?
1932 09743 A2           GOYES   DSPB11          Yes. End output.
1933 09745 E5           B=B+1   A               Counter for #chars (DSPMCW).
1934 09747 876          ?ST=1   width?          Observe width?
1935 0974A 81           GOYES   DSPBF5          Yes. Count only, no display.
1936 0974C 06    DSPBF3 RSTK=C                  Save counter/match in RSTK.
1937 0974E 137          CD1EX
1938 09751 108          R0=C                    Save D1 in R0(A).
1939 09754 8E00         GOSUBL  =DSPCHA         Char to display buffer.
           00
1940 0975A 118          C=R0                    D1 back to C(A).
1941 0975D 135          D1=C                    Restore D1 to output string.
1942 09760 07           C=RSTK
1943 09762 171   DSPBF5 D1=D1+   2               To next char.
1944 09765 52D          GONC    DSPBF1          (BET)
1945                *
1946                *
1947 09768 CE    DSPBF9 C=C-1    A               Count chars.
1948 0976A 51E          GONC    DSPBF3          More to output.
1949 0976D 866   DSPB11 ?ST=0    width?          Observe width?
1950 09770 81           GOYES   DSPB15          No.
1951                *
1952 09772 133          AD1EX                   Yes. A(A)= end addr.
1953 09775 E0           A=A-B   A               B(A)=#chars in string.
1954 09777 E0           A=A-B   A
1955 09779 133          AD1EX                   D1= start addr.
1956 0977C D4           A=B     A               A=#chars in string.
1957 0977E 8F00         GOSBVL  =SENDWD         Display string.
           000
1958 09785 118          C=R0                    To preserve R0.
```

```
1959            *
1960 09788 0B    DSPB15 CSTEX
1961 0978A 80FA         CPEX   10              Restore S7-S4
1962 0978E 80F1         CPEX   1                 from R0(10).
1963 09792 0B           CSTEX
1964 09794 74B3         GOSUB  CSRWP9           Restore R0(A).
1965 09798 108          R0=C                    Only R0(10-5) used.
1966 0979B 03           RTNCC
1967            *
```

```
1968                    EJECT
1969                ********************************************************************
1970                ********************************************************************
1971                **
1972                ** Name:(S) LXFND   -  Set D1 to LEX Table I/O Buffer
1973                **
1974                ** Category:   BUFUTL
1975                **
1976                ** Purpose:
1977                **      Set D1 to LEX table I/O buffer.
1978                **
1979                ** Entry:
1980                **      no necessary conditions.
1981                **
1982                ** Exit:
1983                **      P      =  0
1984                **      Carry set: buffer found.
1985                **        A(A)= buffer length
1986                **        D1 points past buffer header.
1987                **        C(S)=#addresses to update in buffer (?=0)
1988                **      Carry clear: buffer not found.
1989                **
1990                ** Calls:      I/OFND
1991                **
1992                ** Uses.......
1993                **  Exclusive: C(X), P
1994                **  Inclusive: A,C(A),C(S),D1
1995                **
1996                ** Stk lvls:   0
1997                **
1998                **
1999                ** History:
2000                **
2001                **    Date     Programmer              Modification
2002                **  --------   ----------   ----------------------------------
2003                **  01/05/83   MB           Documentation
2004                **
2005                ********************************************************************
2006                ********************************************************************
2007 0979D 20      =LXFND  P=      0
2008 0979F 32CF            LC(3)  =bLEX          LEX buffer I/O number.
           B
2009 097A4 8D00 =i/ofnd GOVLNG =I/OFND           Point D1 to LEX tbl directory.
           000
2010                *
```

```
2011                    EJECT
2012                ****************************************************************
2013                ****************************************************************
2014                **
2015                ** Name:    LERRM   -  Last Error Message
2016                **
2017                ** Category:   FNEXEC
2018                **
2019                ** Purpose:
2020                **      Execute g-Space function
2021                **
2022                ** Entry:
2023                **      No necessary conditions
2024                **
2025                ** Exit:
2026                **      Exits to MAIN10
2027                **
2028                ** Calls:      STAKUP, MsgAVS, VIEWD1, STAKDN
2029                **
2030                ** Uses.......
2031                **  Exclusive: P, A(W), B(W), C(W), D(W), D0, D1, R0
2032                **  Inclusive: same
2033                **
2034                ** Stk lvls:   3
2035                **
2036                ** Algorithm:
2037                **      If in CALC mode, restore true AvMem configuration.
2038                **      Fetch ERRN, set D0= AvMemStart.
2039                **      Build Message text in AvMemStart buffer.
2040                **      If message < 22 bytes, zero additional bytes up to 22.
2041                **      Call VIEWD1 to display error message without disturbing
2042                **         display buffer.
2043                **      If in CALC mode, restore CALC AvMem configuration.
2044                **      Exit.
2045                **
2046                ** History:
2047                **
2048                **    Date     Programmer              Modification
2049                **   --------   ----------   ------------------------------------
2050                **   06/25/82   MB           Wrote routine
2051                **   01/18/83   MB           CALC mode memory moving
2052                **
2053                ****************************************************************
2054                ****************************************************************
2055                *
2056 097AB         =LERRM                          Last Error Message (g-Space).
2057 097AB 8E00            GOSUBL =fCALC?          In CALC mode?
            00
2058 097B1 590             GONC   LERRM1           No.
2059 097B4 8F00            GOSBVL =STAKUP          Yes. Restore AvMem configuration.
            000
2060 097BB 7ED1   LERRM1   GOSUB  MsgAVS           Build msg in AvMemSt.
2061 097BF 131             D1=A                    To D1 for VIEWD1.
2062 097C2 EA              A=A-C  A                A(A)= -#nibs in message.
2063 097C4 CC              A=A-1  A                In case A(A)=0, sets carry below.
```

```
2064 097C6 136          CDOEX               DO--> FF terminator.
2065 097C9 32B2         LC(3)  43           #nibs for window of 22 chars.
           0
2066 097CE A3A          A=A+C  X            A(B)= #nibs to null out.
2067 097D1 511          GONC   LERRM5       NC= message > 44 nibs already.
2068 097D4 D2           C=0    A
2069 097D6 81C          ASRB                A(B)= #bytes to clear.
2070            *
2071 097D9 7A31  LERRM3 GOSUB  DOASub       Zero a byte in buffer.
2072 097DD A6C          A=A-1  B            Count #nibs to clear.
2073 097E0 58F          GONC   LERRM3
2074            *
2075 097E3 8F00  LERRM5 GOSBVL =VIEWD1      View message.
           000
2076 097EA 8E00         GOSUBL =fCALC?      In CALC mode?
           00
2077 097F0 500          RTNNC               No. Return to key read.
2078 097F3 8D00         GOVLNG =STAKDN      Yes. Restore CALC AvMem config,
           000
2079            *                             return to key read.
2080            *
```

```
2081                    EJECT
2082            ****************************************************************
2083            ****************************************************************
2084            **
2085            ** Name:    ERRM$   -  Error Message Function
2086            **
2087            ** Category:   FNEXEC
2088            **
2089            ** Purpose:
2090            **      Execute ERRM$ function.
2091            **
2092            ** Entry:
2093            **      P      = 0
2094            **      D1= stack address,  D0= PC address
2095            **
2096            ** Exit:
2097            **      P      = 0
2098            **      D1 = new stack pointer
2099            **      ERRM$ string on stack
2100            **      D0 unchanged
2101            **
2102            ** Calls:  R3=D1C, MsgAVS, D1C=R3, BF2ST+
2103            **
2104            ** Uses.......
2105            **  Exclusive: B,C
2106            **  Inclusive: A,B,C,D,D1,P,R3,R1
2107            **
2108            ** Stk lvls:   2
2109            **
2110            ** NOTE:
2111            **      See ERRM$f heading for that entry point.
2112            **
2113            ** Algorithm:
2114            **      Entry D1 (stack pointer) and D0 (PC) are saved in R3.
2115            **      Message is built starting at AvMemSt.
2116            **      D1 and D0 are restored from R3.
2117            **      Before calling BF2ST+, which moves the message from
2118            **        AvMemSt to the math stack, checks whether total
2119            **        available memory is at least twice as large as the
2120            **        length of the message (since copying it to the
2121            **        stack would otherwise overwrite the tail end of
2122            **        of the message).  If not, MEMERR.
2123            **      Exits through BF2ST+: buffer to math stack.
2124            **
2125            ** History:
2126            **
2127            **    Date     Programmer          Modification
2128            **  --------   ----------   ----------------------------------
2129            **  09/14/82   MB           Documentation
2130            **
2131            ****************************************************************
2132            ****************************************************************
2133
2134            ****************************************************************
2135            ****************************************************************
```

```
2136                **
2137                ** Name:(S) ERRM$f  -  Transfer ASCII from AvMem to stack
2138                **
2139                ** Category:   MTHSTK
2140                **
2141                ** Purpose:
2142                **      Transfer an ASCII buffer from AvMemSt to Math Stack.
2143                **
2144                ** Entry:
2145                **      P      = 0
2146                **      R3(A)= PC address (from D0)  (see R3=D10)
2147                **      R3(9-5)= stack address (from D1)  (see R3=D10)
2148                **      D0 points to ASCII buffer. ASCII string ends
2149                **        in FF byte. (D0 must be less than FORSTK pointer.)
2150                **      B(A) points to terminator FF byte
2151                **
2152                ** Exit:
2153                **      P      = 0
2154                **      D1 = new stack pointer
2155                **      String on stack
2156                **      D0 = address passed in R3(A)
2157                **      Will jump to MEMERR if insufficient memory.
2158                **
2159                ** Calls:  D1C=R3, BF2ST+
2160                **
2161                ** Uses.......
2162                **  Exclusive: B(A)
2163                **  Inclusive: A(W),B(A),C(A),D(A),R1,D1
2164                **
2165                ** Stk lvls:   1
2166                **
2167                ** NOTE:
2168                **      See ERRM$ heading for that entry point.
2169                **
2170                ** Algorithm:
2171                **      D1 and D0 are restored from R3.
2172                **      Before calling BF2ST+, which moves the message from
2173                **        AvMem to the math stack, checks whether total
2174                **        available memory is at least twice as large as the
2175                **        length of the string (since copying it to the
2176                **        stack would otherwise overwrite the tail end of
2177                **        of the string).  If not, MEMERR.
2178                **      Exits through BF2ST+: buffer to math stack.
2179                **
2180                ** History:
2181                **
2182                **    Date     Programmer             Modification
2183                **  --------   ----------   ------------------------------------
2184                **  09/14/82   MB           Documentation
2185                **
2186                ****************************************************************
2187                ****************************************************************
2188                *
2189 097FA 00           NIBHEX 00              No argument for ERRM$.
2190 097FC 8E00 =ERRM$  GOSUBL =R3=D10         Save D1&D0 in R3 (preserves cry).
```

```
            00
2191 09802 7791          GOSUB  MsgAVS        Build message in AvMemSt.
2192 09806 8E00 =ERRM$f  GOSUBL =D1C=R3       Restore D1; C= old D0 (Also A=D1)
            00
2193 0980C 136           CD0EX                C=start of ASCII; D0 restored.
2194             *                            Transfer of ASCII buff to stack
2195 0980F C5            B=B+B  A               requires RAM= 2*buffer length.
2196 09811 E1            B=B-C  A             Min stack addr for full transfer.
2197 09813 8BC           ?A<=B  A             ASCII buffer conflict with stack?
2198 09816 90            GOYES  DOASER        Yes; MEMERR.
2199 09818 8D00 =bf2st+  GOVLNG =BF2ST+       Put in stack.
            000
2200             *
```

```
2201                    EJECT
2202            ********************************************************************
2203            ********************************************************************
2204            **
2205            ** Name:(S) DOASCI  -  Send ASCII bytes to DATO
2206            ** Name:(S) DOASC+  -  Send ASCII bytes to DATO
2207            **
2208            ** Category:   GENUTL
2209            **
2210            ** Purpose:
2211            **      Build a buffer of ASCII characters starting at D0;
2212            **      the ASCII characters can originate from four types:
2213            **         1)  BCD digits
2214            **         2)  HEX digits
2215            **         3)  numeric conversion from Hex-to-Dec
2216            **         4)  existing ASCII bytes (or tokens)
2217            **      Output can reside in one of two places:
2218            **         1)  in B register
2219            **         2)  in DAT1
2220            **
2221            ** Entry:
2222            **   D0= output address (must be less than AVMEME pointer)
2223            **
2224            **   B register or D1: source of text insertion.
2225            **   C(1):  type of insertion.
2226            **   C(0):  how many characters in insertion.
2227            **
2228            **      B
2229            **      -----
2230            **        = actual output characters or digits
2231            **             if C(1)= 1xxx
2232            **        = additionally, if C(1)= 0000, upper byte
2233            **             of B contains control nibbles.
2234            **
2235            **      D1
2236            **      -----
2237            **        = address of output characters if C(1)= 0xxx
2238            **
2239            **      C(1)
2240            **      -----
2241            **       1xxx   use contents of B register as output
2242            **       0xxx   use address in D1 to find output
2243            **
2244            **       x000   Output is already in ASCII form
2245            **
2246            **           Digit output (digits can be Hex or Dec):
2247            **       x001   Digit output-- replace leading 0's with blanks
2248            **       x010   Digit output-- don't suppress leading 0's
2249            **       x011   Digit output-- suppress leading 0's
2250            **
2251            **
2252            **           Hex-to-Dec conversions always generate
2253            **             decimal numbers with 7 digits:
2254            **       x100   Hex-to-Dec: suppress up to 3 leading 0's
2255            **       x101   Hex-to-Dec: suppress up to 4 leading 0's
```

```
2256            **       x110   Hex-to-Dec: suppress up to 5 leading 0's
2257            **       x111   Hex-to-Dec: suppress up to 6 leading 0's
2258            **
2259            **
2260            **      C(0)
2261            **      -----
2262            **        For C(1)= 1000 ("ASCII output is in B")
2263            **            C(0)= #nibbles-1 to be output. Hence the
2264            **                  #nibs MUST be even!!; C(0) odd. E.g.,
2265            **                  if 5 chars for output, C(0)=9.
2266            **
2267            **        For C(1)= x0xx (hex or dec digit output)
2268            **            C(0)= #digits-1 to be output, hence
2269            **                  no more than 16.
2270            **
2271            **        For C(1)= x1xx (hex-to-dec conversion)
2272            **            C(0)= #digits-1 in number to be converted
2273            **                  Max hex value for conversion is FFFFF
2274            **                  (1048575 dec), hence C(0) must be 4
2275            **                  or less.
2276            **
2277            **        For C(1)= 0000 ("ASCII output from DAT1")
2278            **            C(0)= 0: no output
2279            **                  1: Send out specified number of
2280            **                     character; B(15-14)= #chars-1.
2281            **                  2: Send out chars until ASCII termin-
2282            **                     ator is found.  ASCII terminator
2283            **                     is passed in B(15-14) (usually
2284            **                     an FF terminator, but any byte
2285            **                     value can be used).
2286            **
2287            ** Entry for DOASC+:
2288            **      This entry point is for "ASCII output from DAT1"
2289            **      only:
2290            **         D1 points to output already in ASCII form
2291            **         C(15-14)= #bytes to output
2292            **      DOASC+ then sets C(B)=01 for appropriate codes.
2293            **
2294            ** Exit:  (May exit through MEMERR if not enough memory)
2295            **      Carry clear
2296            **      P      =  0
2297            **      B(A) = # bytes left in available memory past buffer.
2298            **      D0 points to FF terminator, ready for another call.
2299            **
2300            ** Calls:      HEXDEC (only for hex-to-dec conversion;
2301            **                 i.e., only if C(1)=x1xx)
2302            **             MOVEU3 (only for ASCII output from DAT1;
2303            **                 i.e., only if C(1)=0000)
2304            **
2305            ** Uses.......
2306            **      P, A(W), B(W), C(W), D(15-13)
2307            **      D0
2308            **      Uses D1 only if C(1)=0 (i.e., only if ASCII output
2309            **           from DAT1; otherwise D1 not changed). And then,
2310            **           D1 is only moved past source ASCII.
```

```
2311            **
2312            ** Stk lvls:   1
2313            **
2314            ** Detail:
2315            **    Fills DAT0 with characters from B register or from DAT1
2316            **      (as specified by calling routine). An FF terminator
2317            **      is placed at the end of the buffer, ready for a call
2318            **      to BF2DSP or BF2STK.
2319            **    AvMemEnd is checked for sufficient memory.  This is
2320            **    why D0 at entry must be less than AvMemEnd.
2321            **
2322            **    If ASCII output from DAT1, maximum #characters is 255.
2323            **    If digit output, maximum number of digits is 16.  If
2324            **      ASCII from B, maximum number of characters is 8.
2325            **
2326            **    If source is HEX or BCD digits, converts to ASCII
2327            **      equivalents first, for output to DAT0.
2328            **
2329            **    For numeric Hex-to-Dec output, conversion to BCD is
2330            **      performed, then converted to ASCII for output to
2331            **      DAT0.
2332            **
2333            ** Algorithm:
2334            **      Copy control nibs from C to D, calculate
2335            **        #bytes in AvMEM.
2336            **      Do:
2337            **        If ASCII output, copy bytes to DAT0.
2338            **        If Hex-to-Dec, call to HEXDEC, then digit output.
2339            **        If Digit output, convert digits to ASCII and output.
2340            **      As chars are output, decrement #bytes in AvMEM.
2341            **      Terminate buffer with FF.
2342            **
2343            ** History:
2344            **
2345            **    Date     Programmer              Modification
2346            **  --------   ----------   ------------------------------------
2347            **  06/25/82   MB           Documentation
2348            **
2349            ********************************************************************
2350            ********************************************************************
2351 0981F 6D2C  DOASER GOTO   MEMERR         Out of memory.
2352            ******************************
2353            *
2354 09823 170   DOASC- D1=D1+ 1              Entry for ASCII output from D1.
2355 09826 816   DOASC* CSRC
2356 09829 BF6         CSR    W              Char counter to C(15,14).
2357 0982C AF5  =DOASC+ B=C    W              Counter to B(15,14).
2358 0982F D2          C=0    A              C(B)=01:
2359 09831 E6          C=C+1  A                "Output from D1, count chars."
2360            *
2361 09833 04   =DOASCI SETHEX
2362 09835 80D0        P=C    0              P= #chars-1 to output.
2363 09839 BF3         DSL    W
2364 0983C BF3         DSL    W
2365 0983F BF3         DSL    W              Save all of D except D(15-13).
```

```
2366 09842 AE7          D=C    B
2367 09845 BB3          DSL    X              C(XS)= output codes.
2368 09848 AE7          D=C    B              Save again in D(B).
2369 0984B 132          ADOEX
2370              *
2371 0984E 1B99         DO=(5) =AVMEME        Check for insuff memory.
           5F2
2372 09855 AF2          C=0    W              For CSRB below.
2373 09858 146          C=DAT0 A
2374              *
2375 0985B 130          DO=A                  Restore DO.
2376 0985E CE           C=C-1  A              To check two nibs past output
2377 09860 CE           C=C-1  A                buff (needed for FF terminator)
2378 09862 E2           C=C-A  A              C(A)= #nibs left in avail mem.
2379 09864 4AB          GOC    DOASER         MEM ERR.
2380 09867 81E          CSRB                  #bytes in avail mem.
2381 0986A 92B          ?D=0   XS             ASCII output from D1?
2382 0986D E3           GOYES  DOA63j         Yes.
2383 0986F AF0          A=0    W
2384 09872 A94          A=B    WP             Output from B.
2385 09875 A27          D=D+D  XS             Use DAT1 instead?
2386 09878 460          GOC    DOAS05         No.
2387 0987B 1531         A=DAT1 WP             Yes.
2388 0987F 20    DOAS05 P=     0
2389 09881 92B          ?D=0   XS             ASCII output?
2390 09884 B2           GOYES  DOA41j         Yes.
2391 09886 06           RSTK=C                Save #bytes AvMem in RSTK.
2392 09888 A27          D=D+D  XS             Hex to Dec?
2393 0988B 562          GONC   DOAS09         No.
2394 0988E 8E00         GOSUBL =HEXDEC        Yes. Convert to Dec, into A.
           00
2395 09894 04           SETHEX
2396 09896 3263         LCHEX  636            Mask.
           6
2397 0989B 0E63         D=C&D  B              D(1)= # 0's to suppress-3.
2398 0989F A63          D=C+D  B              D(1)= # 0's to suppress.
2399 098A2 BB3          DSL    X
2400 098A5 A87          D=C    P              D(0)=6= "# of chars to output"
2401              *    At this point, C(S)=0 means "Don't replace ldg 0 w/blank".
2402 098A8 522          GONC   DOAS21         (BET)
2403              *
2404 098AB 62A0  DOA63j GOTO   DOAS63
2405 098AF 447   DOA41j GOC    DOAS41         (BET) from above.
2406              *
2407 098B2 A8B   DOAS09 C=D    P
2408 098B5 F2           CSL    A
2409 098B7 F2           CSL    A              #digits-1 to C(XS).
2410 098B9 A27          D=D+D  XS             Replace ldg 0's with blanks?
2411 098BC 450          GOC    DOAS11         No.
2412 098BF A4E          C=C-1  S              Yes. C(S)=F for flag.
2413 098C2 A27   DOAS11 D=D+D  XS             Suppress leading zeroes?
2414 098C5 550          GONC   DOAS21         No; leave D(XS)=0.
2415 098C8 AA7          D=C    XS             #zeroes to suppress.
2416              *
2417 098CB 814   DOAS21 ASRC                  Left justify in A.
```

```
2418 098CE A2E          C=C-1  XS
2419 098D1 59F          GONC   DOAS21
2420              *
2421 098D4 07           C=RSTK                 C(A)= #bytes AvMem.
2422 098D6 D5           B=C    A               B(A)= AvMem byte counter.
2423              *
2424 098D8 3293  DOAS27 LCHEX  439             ASCII conversion codes.
           4
2425 098DD 810          ASLC                   Next output digit to A(0).
2426 098E0 908          ?A=0   P
2427 098E3 F0           GOYES  DOAS29          Not a zero.
2428 098E5 AA3          D=0    XS              "No more zero suppress."
2429 098E8 98A          ?A<=C  P               ASCII A through F?
2430 098EB 70           GOYES  DOAS29          No.
2431 098ED B0A          A=A-C  P               Hex only: A to F
2432 098F0 F6           CSR    A               Hex only: C(1)=4.
2433 098F2 A86   DOAS29 C=A    P               C(B)= ASCII code.
2434 098F5 92B          ?D=0   XS              Zero suppress?
2435 098F8 E0           GOYES  DOAS31          No. Send out ASCII code.
2436 098FA A2F          D=D-1  XS              Count #zeroes to suppress.
2437 098FD 94A          ?C=0   S               Replace ldg zero with blank?
2438 09900 A0           GOYES  DOAS37          No.
2439 09902 3102         LCASC  \ \             Yes.
2440 09906 7D00  DOAS31 GOSUB  DOASub          To next output byte.
2441 0990A A0F   DOAS37 D=D-1  P               More digits for output?
2442 0990D 5AC          GONC   DOAS27          Yes.
2443 09910 4B2          GOC    DOAS49          (BET)
2444              *
2445 09913 6B0F  DOASEj GOTO   DOASER
2446              *
2447 09917 CD    DOASub B=B-1  A               Count AvMem bytes.
2448 09919 49F          GOC    DOASEj          CRY= MEM ERR.
2449 0991C 14C   DOASu+ DATO=C B               Write out character.
2450 0991F 161          DO=DO+ 2
2451 09922 03           RTNCC
2452              *
2453 09924       DOAS41                        ASCII (or Token) output.
2454 09924 D5           B=C    A               #bytes in avail mem.
2455 09926 AF6          C=A    W               Output chars to C.
2456 09929 7AEF  DOAS43 GOSUB  DOASub          Chars out.
2457 0992D BF6          CSR    W
2458 09930 BF6          CSR    W               Next char to C(B).
2459 09933 A0F          D=D-1  P               Count chars.
2460 09936 A0F          D=D-1  P
2461 09939 5FE          GONC   DOAS43          Loop back for next char.
2462 0993C BF7   DOAS49 DSR    W
2463 0993F BF7          DSR    W               Restore D except for D(15-13).
2464 09942 BF7          DSR    W
2465 09945 31FF  DOAS51 LCHEX  FF              String terminator
2466 09949 14C          DATO=C B                  to end of buffer.
2467 0994C 03           RTNCC                  CRY cleared= normal exit.
2468              *
2469 0994E 20    DOAS63 P=     0
2470 09950 AFD          BCEX   W               B= #bytes in avail mem.
2471 09953 A6F          D=D-1  B               Any output?
```

```
2472 09956 45E          GOC    DOAS49         No.
2473 09959 D2           C=0    A
2474 0995B 812          CSLC
2475 0995E 812          CSLC                  C(B)= #chars-1, or ASCII term.
2476 09961 96B          ?D=0   B              Output specified #chars?
2477 09964 71           GOYES  DOAS73         Yes.
2478 09966 AE7          D=C    B              No. Store ASCII terminator in D.
2479              *
2480 09969 14F  DOAS65  C=DAT1 B              Next char.
2481 0996C 171          D1=D1+ 2
2482 0996F 963          ?C=D   B              Terminator?
2483 09972 AC           GOYES  DOAS49         Yes.
2484 09974 7F9F         GOSUB  DOASub         Char out.
2485 09978 50F          GONC   DOAS65         (BET)
2486              *
2487 0997B E6   DOAS73  C=C+1  A              #chars to output.
2488 0997D E1           B=B-C  A              #bytes avail mem.
2489 0997F 439          GOC    DOASEj         MEM ERR.
2490 09982 136          CDOEX
2491 09985 137          CD1EX
2492 09988 136          CDOEX
2493 0998B C6           C=C+C  A              #nibs to output.
2494 0998D 7000         GOSUB  =MOveu3
2495 09991 136          CDOEX
2496 09994 137          CD1EX
2497 09997 136          CDOEX
2498 0999A 51A          GONC   DOAS49         (BET)  Wrap up.
2499              *
```

```
2500                    EJECT
2501            ****************************************************************
2502            ****************************************************************
2503            **
2504            ** Name:     TBMSTX  -  Find and Build Message From Lex Table
2505            ** Name:(S) TBMSG$  -  Find and Build Message From Lex Table
2506            ** Name:     MsgAvs  -  Build message from table, in AvMemSt
2507            **
2508            ** Category:   EXCUTL
2509            **
2510            ** Purpose:
2511            **      Search LEX tables for desired message, and build it
2512            **      into a buffer at D0.
2513            **
2514            ** Entry:
2515            **      MsgAvs -- RAM location ERR# contains desired msg #
2516            **      TBMSTX -- D0 points to buffer to build message.
2517            **                R0(3-2)= LEX ID#, R0(B)= msg #.
2518            **                P= desired value to clear portion of R0.
2519            **
2520            ** Exit:
2521            **      D0 points to FF terminator at end of built msg.
2522            **      P=0, C(B)= FF.
2523            **      Carry cleared.
2524            **
2525            ** Calls:      LXTFND, DOASCI, CSRWP9, CSLWP9, RANGE,
2526            **
2527            ** Uses:
2528            **  Exclusive: A,B,C,D,D1,D0,R0,P,
2529            **             R2 (if msg calls for text insertion)
2530            **  Inclusive: same
2531            **
2532            ** Stk lvls:   2
2533            **
2534            ** Algorithm:
2535            **   MsgAVS  Set D0=AvMemSt
2536            **           Copy ERRN (from ERR#) into C(3-0)
2537            **   TMBSG$  Set P=15 to disallow all text insertions
2538            **   TBMSTX  Save msg number in B
2539            **       (1) Clear R0(WP)
2540            **           Set D1=start of LEX I/O buffer (LXFND)
2541            **           If message is from LEX ID=00, go to (3).
2542            **       (2) Chain through buffer until:
2543            **              End of buffer: Send out null (msg #0000)
2544            **              LEX buffer match.
2545            **           Compute offset to LEX file message table.
2546            **           Check message table range; if no match,
2547            **              go to (2).
2548            **           (Range match:)
2549            **           Save address of table in D(A).
2550            **       (3) If searching for table title, set message
2551            **              number=00
2552            **           Search table for message number
2553            **           If no match, send out null (msg #0000)
2554            **           (Message match:)
```

```
2555                **      (4) Process cells:
2556                **            If cell id = C, go to (5).
2557                **            If cell id < B, then call DOASCI
2558                **              to output #chars. Process next cell.
2559                **            If cell id = B, then read next nib,
2560                **              call DOASCI. Process next cell.
2561                **            If cell id = D, store present table
2562                **              address in R0, set D1=address in D(A),
2563                **              go to (3).
2564                **            If cell id = E, set  D1=mainframe table
2565                **              address, store present table
2566                **              address in R0, go to (3).
2567                **            If cell id = F0, set B=new msg number
2568                **              from table, go to (1).
2569                **            If cell id = F1, set B=new msg number
2570                **              from R2, go to (1).
2571                **            If cell id = F2 or F3, fetch codes from
2572                **              R2, store present table address in R0,
2573                **              call DOASCI.  Process next cell.
2574                **      (5) If table address in R0 (from previous cell)
2575                **            set D1=that address, go to (4).
2576                **          Else, fall into D0=AVS, return.
2577                **
2578                ** History:
2579                **
2580                **    Date     Programmer            Modification
2581                **  --------   ----------   ----------------------------------
2582                **  01/05/83   MB           Documentation
2583                **
2584                ********************************************************************
2585                ********************************************************************
2586 0999D 7B81   MsgAVS GOSUB  D0=AVS          Set D0=AvMemSt.
2587 099A1 1F4E          D1=(5) =ERR#          Point to ERRN.
           7F2
2588 099A8 147           C=DAT1 A              Read ERRN.
2589 099AB 2F    =TBMSG$ P=     15             To clear all of R0.
2590                *
2591                *                            Clear R0(WP) (P= 9 from MFERR,
2592 099AD D5    =TBMSTX B=C    A               P= 15 from ERRM$ to disallow
2593                *                            insertions.)
2594 099AF A92    TBMS01 C=0     WP
2595 099B2 80CF   TBMSID C=P     15             Make C(S)=0 for title search.
2596 099B6 108           R0=C                  Save flag in R0(S), re-write R0.
2597                *
2598 099B9 D9            C=B    A              LEX# and msg# to C(A).
2599 099BB F6            CSR    A
2600 099BD F6            CSR    A              LEX# to C(B).
2601 099BF D7            D=C    A              LEX# to D(B).
2602 099C1 78DD          GOSUB  LXFND          Sets P=0. D1 to LEX I/Obfr.
2603 099C5 96B           ?D=0   B              LEX #00 specified?
2604 099C8 85            GOYES  MFMSTX         Yes, use main table.
2605                *
2606 099CA 14F    LXTF05 C=DAT1 B              LEX ID for next entry.
2607 099CD 96A           ?C=0   B              End of table?
2608 099D0 E4            GOYES  TBMNUL         Yes. LEX tbl not found.
```

```
2609 099D2 17A          D1=D1+ =lLXENT      D1 to next entry.
2610 099D5 967          ?C#D   B            Is this desired LEX table?
2611 099D8 2F           GOYES  LXTF05        No.
2612              *
2613 099DA 1C4          D1=D1- =lLXADR      Yes. Back up to address.
2614 099DD 143          A=DAT1 A            Address of LEX main table.
2615 099E0 133          AD1EX               Addr of LEX to D1.
2616 099E3 1C8          D1=D1- =oMSGPT      Back up to msg table pointer.
2617 099E6 15F3         C=DAT1 4            Read in offset to msg table.
2618 099EA 8AA          ?C=0   A            Msg table present?
2619 099ED A1           GOYES  LXTF09       No. Try next LEX entry.
2620 099EF 133          AD1EX               Yes. Add offset...
2621 099F2 CA           A=A+C  A                    to D1.
2622 099F4 133          AD1EX               D1 points to msg range.
2623 099F7 DC           ABEX   A            Save LEX buffer pointer in B,
2624 099F9 147          C=DAT1 A              A(B)= desired msg#. C=range.
2625 099FC 8E00         GOSUBL =Range       Msg# in range?
           00
2626 09A02 DC           ABEX   A            (LEX buf pointer back to A)
2627 09A04 5B0          GONC   LXTF11       Yes, in range.
2628 09A07 131   LXTF09 D1=A                LEX buf pointer back to D1.
2629 09A0A 174          D1=D1+ =lLXADR      D1 to next LEX entry.
2630 09A0D 5CB          GONC   LXTF05       (BET) Try next entry.
2631              *
2632 09A10 173   LXTF11 D1=D1+ 4            Msg located! Skip over range fld.
2633 09A13 137          CD1EX
2634 09A16 D7           D=C    A            Copy address of tbl to D(A).
2635 09A18 137          CD1EX
2636 09A1B 5B0          GONC   TBMS03       (BET)
2637              *
2638 09A1E D1    TBMNUL B=0    A            Send out null string.
2639 09A20 1F00  MFMSTX D1=(5) =LEX:MN      Mainframe table start (LEX ID=0).
           000
2640 09A27 118   TBMS03 C=R0               Read flag into C(S).
2641 09A2A 94E          ?C#0   S            Title search only?
2642 09A2D 40           GOYES  TBMS05       No. Looking for message.
2643 09A2F D1           B=0    A            Yes. Send title (msg number 0).
2644 09A31 D9    TBMS05 C=B    A            C(B)= msg#.
2645              *
2646 09A33 D1           B=0    A
2647 09A35 143   TBMS09 A=DAT1 A            A(B)= msg length, A(3-2)= msg#.
2648 09A38 AE8          B=A    B            Save length to next msg.
2649 09A3B B64          A=A+1  B            End of table?
2650 09A3E 4FD          GOC    TBMNUL       Yes. Send null string.
2651 09A41 F4           ASR    A
2652 09A43 F4           ASR    A            Table msg# to A(B).
2653 09A45 962          ?A=C   B            Is this the desired msg?
2654 09A48 D0           GOYES  TBMS11       Yes.
2655 09A4A 137          CD1EX               Step to
2656 09A4D C9           C=C+B  A                next message.
2657 09A4F 137          CD1EX
2658 09A52 52E          GONC   TBMS09       (BET)
2659              *
2660 09A55 173   TBMS11 D1=D1+ 4            Message found.
2661              *
```

```
2662 09A58 30C   TBMS15 LCHEX  C             "C"= code for "stop output."
2663 09A5B 143          A=DAT1 A             Read msg cell code.
2664 09A5E 98A          ?A<=C  P             Stop, or ASCII output?
2665 09A61 67           GOYES  TBMS23        Yes.
2666 09A63 118          C=R0                 No.
2667 09A66 D8           B=A    A
2668 09A68 F5           BSR    A             B(B)= msg#, or B(0)= insert flag.
2669 09A6A B04          A=A+1  P             Insert text?
2670 09A6D 4D1          GOC    TBMS19        Yes.
2671 09A70 172          D1=D1+ 3             No. Building Block; to next cell.
2672 09A73 7AE0         GOSUB  CSLWP9        Shift R0(A) to nibs 9-5.
2673 09A77 137          CD1EX                Save next msg cell addr in R0(A).
2674 09A7A 108          R0=C
2675 09A7D B04          A=A+1  P             Mainframe building block?
2676 09A80 4F9          GOC    MFMSTX        Yes.
2677 09A83 DB           C=D    A             No. Local msg#.
2678 09A85 135          D1=C                 LEX tbl addr to D1.
2679 09A88 58A          GONC   TBMS05        (BET)
2680             *
2681             *  Special cell codes:  F0= message from different XROM
2682             *                       F1= insert indirect message
2683             *                       F2= insert text with no trailing space
2684             *                       F3= insert text with trailing space
2685             *
2686 09A8B 171   TBMS19 D1=D1+ 2             Insert caller's text.
2687 09A8E A0D          B=B-1  P             Message from different XROM?
2688 09A91 467          GOC    TBMS31        Yes.
2689 09A94 BF2          CSL    W
2690 09A97 D9           C=B    A             Flags to C(0), end up in C(XS).
2691 09A99 812          CSLC
2692 09A9C 812          CSLC                 C(B)= DOASCI codes.
2693 09A9F 96A          ?C=0   B             Any text?
2694 09AA2 6B           GOYES  TBMS15        No. Don't wipe out R2.
2695 09AA4 112          A=R2                 Yes.
2696 09AA7 AF8          B=A    W             Either B= text, or upper B
2697             *                             =codes and DAT1= text.
2698 09AAA A2E          C=C-1  XS            Insert indirect message?
2699 09AAD 426          GOC    TBMS35        Yes.
2700 09AB0 133          AD1EX                No. Insert text.
2701 09AB3 102          R2=A                 Save message addr in R2.
2702 09AB6 92A          ?C=0   XS            Trailing space?
2703 09AB9 01           GOYES  TBMS20        No. Text to buffer only.
2704 09ABB 747D         GOSUB  DOASCI        Output text to buffer.
2705 09ABF 3102         LCASC  \ \           Send trailing blank.
2706 09AC3 D5           B=C    A             Blank to B,
2707 09AC5 3118         LCHEX  81              codes in C.
2708 09AC9 766D  TBMS20 GOSUB  DOASCI
2709 09ACD 7570         GOSUB  CSRW9-        Fetch msg address.
2710 09AD1 10A          R2=C                 2nd insert to R2(A), D1=msg addr.
2711 09AD4 538   TBMS21 GONC   TBMS15        (BET) To next msg cell.
2712             *
2713 09AD7 902   TBMS23 ?A=C   P             Stop output?
2714 09ADA 71           GOYES  TBMS27        Yes.
2715 09ADC CE           C=C-1  A             C(0)= B.
2716 09ADE 982          ?A<C   P             A(0)= #chars-1 to output?
```

```
2717 09AE1 70          GOYES  TBMS25        Yes.
2718 09AE3 170         D1=D1+ 1             No. #chars-1 in next nib.
2719 09AE6 F4          ASR    A             #chars-1 to A(0).
2720 09AE8 D6    TBMS25 C=A    A             C(0)= #chars-1 to output.
2721 09AEA 753D        GOSUB  DOASC-        Text to buffer.
2722 09AEE 55E         GONC   TBMS21        (BET) Next cell.
2723             *
2724 09AF1 118   TBMS27 C=R0                Fetch "return" msg addr.
2725 09AF4 94A         ?C=0   S             Were we just searching for title?
2726 09AF7 00          RTNYES               Yes. Title found and sent.
2727 09AF9 8AA         ?C=0   A             End of complete msg?
2728 09AFC 42          GOYES  TBMS39        Yes.
2729 09AFE 7740        GOSUB  CSRW9+        No. Address of next cell
2730 09B02 108         R0=C                   to R0(A).
2731 09B05 5EC         GONC   TBMS21        (BET)  Next cell.
2732             *
2733 09B08 143   TBMS31 A=DAT1 A             Read new XROM msg number.
2734 09B0B D8          B=A    A             To B.
2735 09B0D 4C0         GOC    TBMS37        (BET) Terminate curr msg.
2736             *
2737 09B10 7230  TBMS35 GOSUB  CSRW9-        Next insert codes to R2(A).
2738 09B14 10A         R2=C
2739 09B17 118         C=R0                  To save R0(15-10).
2740 09B1A 29    TBMS37 P=     9             Terminate current msg,
2741 09B1C 629E        GOTO   TBMS01          call indirect msg#.
2742             *
2743 09B20 7F0D  TBMS39 GOSUB  DOASCI        End of msg. Since C(A)=0, sends
2744             *                             out FF terminator (consider
2745             *                             null message).
2746 09B24 136         CDOEX                 C=addr of FF at end of msg.
2747 09B27 D5          B=C    A             Copy addr to B.
2748 09B29 110         A=R0                  To preserve upper R0.
2749             *
2750             * ! ! ! ! ! ! ! ! Fall into DO=AVS ! ! ! ! ! ! ! !
2751             *
```

```
2752                    EJECT
2753                * ! ! ! ! ! ! ! !  Above code falls into here ! ! ! ! ! ! ! !
2754                ****************************************************************
2755                ****************************************************************
2756                **
2757                ** Name:(S) D0=AVS  -  Set D0=address in AVMEMS
2758                ** Name:(S) D0=PCA  -  Set D0=address in PCADDR
2759                **
2760                ** Category:   PTRUTL
2761                **
2762                ** Purpose:
2763                **      D0=AVS : Set D0=<AVMEMS> (also set A(A)=<AVMEMS>)
2764                **      D0=PCA : Set D0=<PCADDR> (also set A(A)=<PCADDR>)
2765                **
2766                ** Entry:
2767                **      No necessary conditions
2768                **
2769                ** Exit:
2770                **      D0=AVS : D0=A(A)=<AVMEMS>
2771                **      D0=PCA : D0=A(A)=<PCADDR>
2772                **      Carry not affected.
2773                **
2774                ** Calls:      None
2775                **
2776                ** Uses:..... D0, A(A)
2777                **
2778                ** Stk lvls:   none
2779                **
2780                ** Detail:
2781                **          =D0=AVS D0=(5) =AVMEMS    Set D0= start of avail mem.
2782                **                  GOTO   D0=DTO
2783                **          =D0=PCA D0=(5) =PCADDR    Set D0= addr of xqtn line.
2784                **           D0=DTO A=DATO A          Set D0= address in DATO.
2785                **                  D0=A
2786                **                  RTN
2787                **
2788                ** History:
2789                **
2790                **    Date     Programmer              Modification
2791                **  --------   ----------   ----------------------------------
2792                **  01/05/83   MB           Documentation
2793                **
2794                ****************************************************************
2795                ****************************************************************
2796                * ! ! ! ! ! ! ! ! Above code falls into here ! ! ! ! ! ! ! !
2797
2798 09B2C 1B49 =D0=AVS D0=(5) =AVMEMS        Set D0= start of avail mem.
          5F2
2799 09B33 6A00          GOTO   D0=DTO
2800 09B37 1B97 =D0=PCA D0=(5) =PCADDR        Set D0= address of execution line.
          6F2
2801 09B3E 142   D0=DTO A=DATO A              Set D0= address in DATO.
2802 09B41 130           D0=A
2803 09B44 01            RTN
2804                *
```

```
2805                    EJECT
2806                ****************************************************************
2807                ****************************************************************
2808                **
2809                ** Name:    CSRWP9  -  Do five CSR WP with P=9
2810                ** Name:    CSRWP   -  Do five CSR WP, P as passed
2811                **
2812                ** Category:   GENUTL
2813                **
2814                ** Purpose:
2815                **      Shift C right 5 nibbles with P=9.
2816                **
2817                ** Entry:
2818                **      No necessary conditions
2819                **
2820                ** Exit:
2821                **      P       =  0
2822                **      Carry cleared
2823                **
2824                ** Uses        P, C(9-0)
2825                **
2826                ** Stk lvls:   None
2827                **
2828                ** Detail:
2829                **      =CSRWP9 P=      9               C(9-5) to C(A).
2830                **      =CSRWP  CSR     WP
2831                **              CSR     WP
2832                **              CSR     WP
2833                **              CSR     WP
2834                **              CSR     WP
2835                **              P=      0
2836                **              RTNCC
2837                **
2838                ** History:
2839                **
2840                **    Date      Programmer              Modification
2841                **  --------    ----------    ---------------------------------
2842                **  06/25/82    MB            Documentation
2843                **
2844                ****************************************************************
2845                ****************************************************************
2846 09B46 11A     CSRW9-  C=R2
2847 09B49 135     CSRW9+  D1=C
2848 09B4C 29     =CSRWP9  P=      9               C(9-5) to C(A).
2849 09B4E B96    =CSRWP   CSR     WP
2850 09B51 B96             CSR     WP
2851 09B54 B96     CSRWP*  CSR     WP
2852 09B57 B96             CSR     WP
2853 09B5A B96             CSR     WP
2854 09B5D 20              P=      0
2855 09B5F 03      Mrtncc  RTNCC
2856                *
```

```
2857                    EJECT
2858                ****************************************************************
2859                ****************************************************************
2860                **
2861                ** Name:    CSLWP9  -  Do five CSL WP with P=9
2862                ** Name:    CSLWP   -  Do five CSL WP, with P as passed
2863                **
2864                ** Category:   GENUTL
2865                **
2866                ** Purpose:
2867                **      Shift C left 5 nibbles with P=9.
2868                **
2869                ** Entry:
2870                **      No necessary conditions
2871                **
2872                ** Exit:
2873                **      P       =  0
2874                **      Carry cleared
2875                **
2876                ** Uses        P, C(9-0)
2877                **
2878                ** Stk lvls:   None
2879                **
2880                ** Detail:
2881                **      =CSLWP9 P=      9               C(A) to C(9-5).
2882                **      =CSLWP  CSL     WP
2883                **              CSL     WP
2884                **              CSL     WP
2885                **              CSL     WP
2886                **              CSL     WP
2887                **              P=      0
2888                **              RTNCC
2889                **
2890                ** History:
2891                **
2892                **    Date      Programmer              Modification
2893                **  --------    ----------   ---------------------------------
2894                **  06/25/82    MB           Documentation
2895                **
2896                ****************************************************************
2897                ****************************************************************
2898 09B61 29   =CSLWP9 P=      9
2899 09B63 B92  =CSLWP  CSL     WP
2900 09B66 B92          CSL     WP
2901 09B69 B92          CSL     WP
2902 09B6C B92          CSL     WP
2903 09B6F B92          CSL     WP
2904 09B72 20           P=      0
2905 09B74 03           RTNCC
2906                *
2907 09B76              END
```

```
 A=CUR   Ext                  -  1479
 ATCHK   Ext                  -  1516
 AUTCLR  Ext                  -  1353
 AVMEME  Abs  193945 #2F599   -    14  2371
 AVMEMS  Abs  193940 #2F594   -    14  2798
=AVS2DS  Abs   38664 #09708   -  1906  1489
 AVS=C   Ext                  -  1485  1488
 BF2ST+  Ext                  -  2199
 BLDDSP  Ext                  -  1497
=BSERR   Abs   37786 #0939A   -   595   806
 BSTYCK  Ext                  -  1733
 CHIRP   Ext                  -  1492
 CKINF-  Ext                  -  1652
 CLCOLL  Ext                  -  1352
 COLLAP  Ext                  -  1351
 CPL#10  Ext                  -  1736
 CRLFND  Ext                  -  1373  1486
 CRLFSD  Ext                  -  1496
=CSLWP   Abs   39779 #09B63   -  2899
=CSLWP9  Abs   39777 #09B61   -  2898  1150  1925  2672
 CSRW9+  Abs   39753 #09B49   -  2847  2729
 CSRW9-  Abs   39750 #09B46   -  2846  2709  2737
=CSRWP   Abs   39758 #09B4E   -  2849
 CSRWP*  Abs   39764 #09B54   -  2851
=CSRWP9  Abs   39756 #09B4C   -  2848  1675  1964
 CURRL   Abs  194536 #2F7E8   -    14  1454
 CURSFL  Ext                  -  1677
 CURSRR  Ext                  -  1681
=CURSRT  Abs   38593 #096C1   -  1676
=DO=AVS  Abs   39724 #09B2C   -  2798  1439  1906  2586
 DO=DTO  Abs   39742 #09B3E   -  2801  1663  2799
=DO=PCA  Abs   39735 #09B37   -  2800  1514  1735
 DOA41j  Abs   39087 #098AF   -  2405  2390
 DOA63j  Abs   39083 #098AB   -  2404  2382
 DOAS05  Abs   39039 #0987F   -  2388  2386
 DOAS09  Abs   39090 #098B2   -  2407  2393
 DOAS11  Abs   39106 #098C2   -  2413  2411
 DOAS21  Abs   39115 #098CB   -  2417  2402  2414  2419
 DOAS27  Abs   39128 #098D8   -  2424  2442
 DOAS29  Abs   39154 #098F2   -  2433  2427  2430
 DOAS31  Abs   39174 #09906   -  2440  2435
 DOAS37  Abs   39178 #0990A   -  2441  2438
 DOAS41  Abs   39204 #09924   -  2453  2405
 DOAS43  Abs   39209 #09929   -  2456  2461
 DOAS49  Abs   39228 #0993C   -  2462  2443  2472  2483  2498
 DOAS51  Abs   39237 #09945   -  2465
 DOAS63  Abs   39246 #0994E   -  2469  2404
 DOAS65  Abs   39273 #09969   -  2480  2485
 DOAS73  Abs   39291 #0997B   -  2487  2477
 DOASC*  Abs   38950 #09826   -  2355  1450
=DOASC+  Abs   38956 #0982C   -  2357
 DOASC-  Abs   38947 #09823   -  2354  2721
=DOASCI  Abs   38963 #09833   -  2361  1465  2704  2708  2743
 DOASER  Abs   38943 #0981F   -  2351  2198  2379  2445
 DOASEj  Abs   39187 #09913   -  2445  2448  2489
```

```
 DOASu+  Abs   39196 #0991C -  2449  1460  1467
 DOASub  Abs   39191 #09917 -  2447  2071  2440  2456  2484
 DOOUTB  Ext                -  1914
 D1C=R3  Ext                -  2192
=DONNA   Abs   38486 #09656 -  1646
 DSPB11  Abs   38765 #0976D -  1949  1932
 DSPB15  Abs   38792 #09788 -  1960  1950
 DSPBF1  Abs   38712 #09738 -  1928  1944
 DSPBF3  Abs   38732 #0974C -  1936  1948
 DSPBF5  Abs   38754 #09762 -  1943  1935
 DSPBF9  Abs   38760 #09768 -  1947  1930
=DSPBUF  Abs   38691 #09723 -  1920  1659  1910
 DSPCHA  Ext                -  1673  1939
=DSPCNA  Abs   38689 #09721 -  1917  1671
=DSPCNB  Abs   38687 #0971F -  1916
=DSPCNO  Abs   38678 #09716 -  1914
 ERR#    Abs  194532 #2F7E4 -    14  1408  2587
 ERRADR  Abs  194184 #2F688 -    14  1433
=ERRM$   Abs   38908 #097FC -  2190
=ERRM$f  Abs   38918 #09806 -  2192
 ERRMST  Ext                -  1445
 ERRRTN  Ext                -   599
 ERRSUB  Abs  194179 #2F683 -    14  1429  1433
 ESC"<"  Abs   38619 #096DB -  1686  1374  1647
 ESCSEQ  Ext                -  1688
 ESCSEq  Abs   38625 #096E1 -  1688  1661
 FPoll   Ext                -  1346
 HEXDEC  Ext                -  2394
 I/OFND  Ext                -  2009
 INBS    Abs  194246 #2F6C6 -    14  1143  1662
 InhEOL  Abs       4 #00004 -    17    18    19  1655  1917
 KILLKY  Ext                -   813
=LERRM   Abs   38827 #097AB -  2056
 LERRM1  Abs   38843 #097BB -  2060  2058
 LERRM3  Abs   38873 #097D9 -  2071  2073
 LERRM5  Abs   38883 #097E3 -  2075  2067
 LEX:MN  Ext                -  2639
=LXFND   Abs   38813 #0979D -  2007  2602
 LXTFO5  Abs   39370 #099CA -  2606  2611  2630
 LXTFO9  Abs   39431 #09A07 -  2628  2619
 LXTF11  Abs   39440 #09A10 -  2632  2627
=MEMER*  Abs   37979 #0945B -  1341  1336
 MEMER1  Abs   38046 #0949E -  1362  1156  1335
=MEMERR  Abs   37965 #0944D -  1334  2351
=MEMERX  Abs   37967 #0944F -  1335
 MFER+6  Abs   38079 #094BF -  1374  1372
 MFER.6  Abs   38057 #094A9 -  1368  1158  1356
 MFERO2  Abs   37888 #09400 -  1132   815
 MFERO3  Abs   37896 #09408 -  1135  1131
 MFERO4  Abs   37938 #09432 -  1149  1147
 MFERO5  Abs   37951 #0943F -  1154  1138  1140
 MFERO7  Abs   38092 #094CC -  1392  1376
 MFERO8  Abs   38120 #094E8 -  1402  1397
 MFERO9  Abs   38129 #094F1 -  1406  1403
 MFER11  Abs   38164 #09514 -  1417  1407  1411
```

```
 MFER15   Abs    38215 #09547 -   1439  1418  1428  1432  1436
 MFER19   Abs    38252 #0956C -   1449  1447
 MFER25   Abs    38295 #09597 -   1463  1458
 MFER27   Abs    38306 #095A2 -   1466  1453
 MFER29   Abs    38310 #095A6 -   1467  1461
 MFER33   Abs    38314 #095AA -   1469  1441
 MFER35   Abs    38372 #095E4 -   1489  1482
 MFER37   Abs    38387 #095F3 -   1493  1491
 MFER39   Abs    38398 #095FE -   1496  1494
 MFER41   Abs    38437 #09625 -   1507  1506
=MFER42   Abs    38444 #0962C -   1509
 MFER51   Abs    38486 #09656 -   1645  1508
 MFER53   Abs    38520 #09678 -   1656  1651
 MFER55   Abs    38578 #096B2 -   1672  1670
 MFER57   Abs    38603 #096CB -   1679  1682
=MFERR    Abs    37779 #09393 -    592
=MFERR*   Abs    37873 #093F1 -   1126   595
=MFERR-   Abs    37870 #093EE -   1125
=MFERRS   Abs    37790 #0939E -    596  1338
=MFERsp   Abs    37901 #0940D -   1136
 MFLG=X   Ext                  -   1499
 MFMSTX   Abs    39456 #09A20 -   2639  2604  2676
 MFWR03   Abs    37831 #093C7 -    804   800
 MFWR05   Abs    37858 #093E2 -    813   809
=MFWRN    Abs    37820 #093BC -    798
=MFWRNQ   Abs    37829 #093C5 -    803
=MFWRQ8   Abs    37827 #093C3 -    802
 MOveu3   Ext                  -   2494
 Mrtncc   Abs    39775 #09B5F -   2855
 MsgAVS   Abs    39325 #0999D -   2586  2060  2191
 MsgPOL   Abs    37805 #093AD -    605   804  1126
 NoCont   Abs       14 #0000E -     13   596
 ONERR    Ext                  -   1437
 PCADDR   Abs   194169 #2F679 -     14  2800
 POllj    Ext                  -    608
=PRP3>3   Abs    38532 #09684 -   1660  1648
 R3=D10   Ext                  -   2190
 R<RST2   Ext                  -   1368
 RST2<R   Ext                  -   1501
 Range    Ext                  -   2625
 SAVELO   Ext                  -   1741
 SENDWD   Ext                  -   1957
 SFlag?   Ext                  -    811
 STOSW1   Abs    38465 #09641 -   1521
 STOSW3   Abs    38469 #09645 -   1522  1394
 STOSWP   Abs    38462 #0963E -   1520  1504  1507
 STAKDN   Ext                  -   2078
 STAKUP   Ext                  -   2059
 SflagC   Ext                  -   1421
 TBMNUL   Abs    39454 #09A1E -   2638  2608  2650
 TBMS01   Abs    39343 #099AF -   2594  2741
 TBMS03   Abs    39463 #09A27 -   2640  2636
 TBMS05   Abs    39473 #09A31 -   2644  2642  2679
 TBMS09   Abs    39477 #09A35 -   2647  2658
 TBMS11   Abs    39509 #09A55 -   2660  2654
```

```
 TBMS15   Abs   39512 #09A58 -  2662  2694  2711
 TBMS19   Abs   39563 #09A8B -  2686  2670
 TBMS20   Abs   39625 #09AC9 -  2708  2703
 TBMS21   Abs   39636 #09AD4 -  2711  2722  2731
 TBMS23   Abs   39639 #09AD7 -  2713  2665
 TBMS25   Abs   39656 #09AE8 -  2720  2717
 TBMS27   Abs   39665 #09AF1 -  2724  2714
 TBMS31   Abs   39688 #09B08 -  2733  2688
 TBMS35   Abs   39696 #09B10 -  2737  2699
 TBMS37   Abs   39706 #09B1A -  2740  2735
 TBMS39   Abs   39712 #09B20 -  2743  2728
=TBMSG$   Abs   39339 #099AB -  2589
 TBMSID   Abs   39346 #099B2 -  2595  1444
=TBMSTX   Abs   39341 #099AD -  2592  1471
 TRNFCK   Ext                -  1355
 UPDCR1   Abs   38656 #09700 -  1739  1734
 UPDCR3   Abs   38658 #09702 -  1741  1738
=UPDCRL   Abs   38631 #096E7 -  1733  1377
 VIEWD1   Ext                -  2075
 WRNMST   Ext                -  1448
 XDELAY   Ext                -  1495
 bLEX     Abs    3068 #00BFC -    13  2008
 beep     Abs       3 #00003 -    25  1490
=bf2st+   Abs   38936 #09818 -  2199
 cntr?    Abs       5 #00005 -    18  1917  1929
 delay    Abs       0 #00000 -    22  1493
 eMEM     Ext                -  1362
 fCALC?   Ext                -  1371  2057  2076
 flNOFN   Abs     -42 #FFFD6 -    13  1420
 flQIET   Abs      -1 #FFFFF -    13   810
=i/ofnd   Abs   38820 #097A4 -  2009
 lLXADR   Abs       5 #00005 -    13  2613  2629
 lLXENT   Abs      11 #0000B -    13  2609
 oMSGPT   Abs       9 #00009 -    13  2616
 pERROR   Abs     242 #000F2 -    13  1127
 pMEM     Abs     241 #000F1 -    13  1347
 pWARN    Abs     243 #000F3 -    13   805
 parse    Abs       1 #00001 -    23  1404  1505
 sENDx    Abs       1 #00001 -    13   597
 sERROR   Abs       0 #00000 -    13   598
=savel0   Abs   38658 #09702 -  1740
 warn     Abs       2 #00002 -    24  1395  1417  1446  1503
 width?   Abs       6 #00006 -    19  1655  1934  1949
```

Input Parameters

  Source file name is TI&ERD::MS

  Listing file name is TI/ERD:TI:ML::-1

  Object file name is TI%ERD:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS    GGG    &     FFFFF  X   X   QQQQ
 2            *    S   S  G   G  & &    F      X   X  Q    Q
 3            *    S      G      & &    F       X X   Q    Q
 4            *     SSS   G GGG   &     FFFF     X    Q    Q
 5            *        S  G   G  & & &  F       X X   Q  Q Q
 6            *    S   S  G   G  &  &   F      X   X  Q   Q
 7            *     SSS    GGG    && &  F      X   X   QQQ Q
 8            *
 9                          TITLE  File Execution Routines<831216.1614>
10 09B76                   ABS    #09B76
11                         RDSYMB TI%EQU::MS
12                         RDSYMB SB%RAM::MS
13            *
14            *
```

```
15                      STITLE File Execute  (FILXQ^)
16          ********************************************************************
17          ********************************************************************
18          **
19          ** Name:(S) FILXQ^  -  Filename Execute
20          ** Name:(S) FILXQ$  -  Filename Execute For a String Expression
21          **
22          ** Category:   FILUTL
23          **
24          ** Purpose:
25          **      Executes a tokenized file specifier.  Solitary device
26          **      Specifiers of the form ':CARD', ':PORT', ':MAIN', etc,
27          **      are accepted. There are two entry points.
28          **
29          **     FILXQ^:
30          **      Assumes that D0 points to a file specifier in program
31          **      memory.  The file specifier may be a literal or a
32          **      string expression.
33          **
34          **     FILXQ$:
35          **      Assumes that the alleged string has been evaluated and
36          **      is on the Math Stack.  This entry is used by ADDR$ and
37          **      CAT$.
38          **
39          ** Entry:
40          **     FILXQ^:
41          **      D0 at start of file specifier
42          **     FILXQ$:
43          **      D1 points to string expression on top of Match Stack
44          **
45          ** Exit:
46          **    CARRY SET: (both entry points)
47          **      Mainframe-recognizable file specifier found.
48          **      A(W)    =  Blank-padded file name if name present.
49          **              =  0 if only device specifier present, as in
50          **                 ':MAIN', ':PORT', ':CARD'
51          **      D(S)    =  F if no device specified
52          **              =  0 if device is :MAIN
53          **              =  1 if device in :PORT, in which case:
54          **                   D(0) = PORT extender number 0-F
55          **                   D(1) = PORT number          0-4
56          **                   D(B) = FF if no PORT number specified
57          **              =  7 if device is card, in which case:
58          **                   D(B) = 0 if :CARD
59          **                   D(B) = Nonzero if :PCRD
60          **      D0      =  Past file specifier  (FILXQ^ only)
61          **      P       =  0
62          **      If file specifier was a string expression:
63          **       D1 points past the string on the stack
64          **      If file specifier was a literal containing a port#:
65          **       D1 points past the 16 nibble number on the stack
66          **       (AVMEME)=D1
67          **
68          **    CARRY CLEAR:
69          **     FILXQ^:
```

```
70          **      Executed illegal mainframe file name. Either string
71          **      expression or literal name with over 8 characters.
72          **      S7=1 =>     Specifier was string expression, in which
73          **                  case the expression is still on the stack.
74          **                   AVMEME points to the string header
75          **                    D0    points past the tokenized expression.
76          **
77          **      S7=0 =>     Specifier was a literal;  D0 may be restored
78          **                  to the start of the literal by using (STMTD0)
79          **      P      = 0
80          **
81          **     FILXQ$:
82          **      String expression on stack contained an illegal specifier
83          **      S7     = 1
84          **      AVMEME = Value it contained on entry.  May be used to
85          **               preserve the pointer to the string header prior
86          **               to calling FILXQ$.
87          **      P      = 0
88          **
89          **    ERROR EXIT (both entry points):
90          **      Exit to MFERR (eFSPEC) if and only if :PORT is found,
91          **      followed by an illegal port specifier.
92          **
93          **
94          ** Calls:      EXPEXC, FILEP, PDEV, CATCHR, DVCTYP, POLL,
95          **             REVPOP, BLKOK, PRT#P, FINDA, RSTST, SAVEDO,
96          **             CNVWUC, AVE=D1
97          ** Uses........
98          **   Exclusive:  A-D, D1, D0, STMTD0, R0, R1, S1,S2,S7
99          **   Inclusive:  STMTR1 (all of it) -- port spec. as num expr
100         **               R0-R3, all of function scratch -- EXPEXC
101         **
102         **         NOTE:  FILXQ$ entry doesn't use any statement scratch.
103
104         **
105         ** Detail:     D0 on entry to FILXQ^ is a pointer to the start
106         **             of the compiled file specification.  FILXQ^ must
107         **             save D0 in STMTD0, since EXPEXC can use all CPU
108         **             registers and all function scratch RAM.  STMTD0
109         **             will be updated if memory moves.
110         **
111         **             SYNTAX FOR PORT# IS <d[.d[d]]>
112         **
113         **             ASSUMES THAT ALL NON-MAINFRAME DEVICE REFERENCES
114         **             HAVE BEEN TOKENIZED WITH tCOLON.
115         **
116         **             Nibs 2,3,4 of D are zeroed out for TRSFMu
117         **
118         ** Stack lvls: FILXQ$ entry pt - 3
119         **             Otherwise       - 5
120         **
121         ** History:
122         **
123         **   Date      Programmer   Modification
124         **  -------    ----------   ------------
```

```
125             ** 06/29/82   S.W.        Added documentation.
126             ** 07/05/82   S.W.        Modified code to eliminate
127             **                         call to POP1S - lets REV$
128             **                         take care of that.
129             ** 07/27/82   S.W.        Added code to check for tCOLON
130             **                         before assuming string expr.
131             ** 10/21/82   S.W.        Save PC in STMTDO, instead
132             **                         of S-R1-0
133             ** 01/31/83   J.P.        Clear S7 on entry
134             ** 06/28/83   S.W.        Save rtn stack level in R0
135             **                         prior to calling FILEP!.
136             **
137             ********************************************************************
138             ********************************************************************
139             *
140             ********************************************************************
141             ********************************************************************
142             **
143             ** Name:(S) pFILXQ  -  Poll for device to return device ID
144             **
145             ** Category:   POLL
146             **
147             ** Type:       POLL
148             **
149             ** Purpose:
150             **      Polls for dedicated device to intervene to return its id
151             **
152             ** Should poll be "Handled" (return with XM=0)?:
153             **      Yes
154             **
155             ** Meaning of "Handling" Poll (what does code do if handled?):
156             **      Reads device specifier (either as an executed string
157             **      expression off stack or as a literal) and if their
158             **      device is referenced, return device ID in D(S) & D(X)
159             **
160             ** Entry conditions for handler (registers, ST, RAM, etc.):
161             **      Carry clear
162             **      B[A] = Poll number.
163             **      HEX mode.
164             **      P=0.
165             **      R0 contains file name (if any) - <=8 characters
166             **      D0 may be restored prior to filespec, using STMTDO
167             **       (this is generally not useful)
168             **      IF S7=0
169             **        Device specifier is a literal
170             **        D0 points past tCOLON (Poll handler must check
171             **         to ensure that D0 points to tLITRL - if it doesn't
172             **         poll should NOT be handled.
173             **      IF S7=1
174             **        Device specifier is a string on the stack
175             **         (string header pointed to by AVMEME)
176             **        D0 points past the entire file specifier
177             **        A colon was found on the stack, in the appropriate
178             **        position.
179             **
```

```
180             ** Normal exit conditions from handler if handled (ST, RAM,
181             ** registers, etc.):
182             **      Carry clear
183             **      HEX mode.
184             **      XM=0.
185             **      File Name in A (Retrieve from R0 before exit)
186             **      D(S),D(X) set appropriately with device id
187             **      D0 past file specifier
188             **      ADDITIONALLY:
189             **        IF S7=1 on entry, then D1 must point past the string
190             **         on the stack and AVMEME must reflect this.
191             **
192             ** Normal exit conditions from handler if not handled (ST, RAM,
193             ** registers, etc.):
194             **      Carry clear
195             **      HEX mode.
196             **      XM=1.
197             **      R0 must be unaltered from entry.
198             **
199             ** Error exit conditions from handler (POLL only):
200             **      NO ERROR - Instead DON'T HANDLE
201             **
202             ** Available subroutine levels:
203             **      6
204             **
205             ** NOTE:
206             **
207             **
208             ** What registers/RAM may be used if handled?:
209             **      A-D,D1,D0,R1,STMTR1 (all of it), S1,S2
210             **
211             ** What registers/RAM may be used if not handled?:
212             **      A-D, D0, D1, P
213             **      R1,STMTR1 (all of it), S1,S2
214             **
215             ** What registers/RAM may be used if error exit (POLL only)?:
216             **      NO error exit
217             **
218             ** Envisioned application(s):
219             **      So a dedicated device may be referenced analogous to an
220             **      HPIL device, eg > INITIALIZE :HP145XX
221             **
222             ** History:
223             **
224             **    Date     Programmer              Modification
225             **  --------   ----------   ----------------------------------
226             **  04/21/83   S.W.         Added POLL & documentation
227             **
228             ********************************************************************
229             ********************************************************************
230             *
231             *
232 09B76 20  =FILXQ^ P=     0
233             * Save PC in STMTD0
234 09B78 8E00        GOSUBL =SAVED0
```

```
          00
235             *
236 09B7E AF0           A=0    W              NO FILE NAME SPECIFIED
237 09B81 100           R0=A
238 09B84 847           ST=0   7              Clear String Expr flag
239 09B87 7AD1          GOSUB  DVCTYP
240             *
241             * Must be expression
242             *
243 09B8B 8E00          GOSUBL =EXPEX+        Save statuses
          00
244 09B91 7293          GOSUB  rstst          Restore statuses
245 09B95 857  =FILXQ$  ST=1   7              Set string expression flag
246             *
247 09B98 8E00          GOSUBL =REVPOP        REVERSE STRING ON STACK
          00
248             * D1 past header
249 09B9E 8A8           ?A=0   A              NULL STRING?
250 09BA1 D2            GOYES  FLXQ05
251 09BA3 D8            B=A    A              Copy string length
252 09BA5 137           CD1EX
253 09BA8 135           D1=C
254 09BAB C2            C=C+A  A              POINT PAST LAST CHARACTER
255 09BAD AF3           D=0    W              IN PRT#P, WANT NIB5 = 0
256 09BB0 D7            D=C    A              SAVE PTR PAST LAST CHAR IN STRING
257 09BB2 31A3          LCASC  \:\
258 09BB6 14B           A=DAT1 B
259 09BB9 962           ?A=C   B
260 09BBC 54            GOYES  FLXQ11
261             *
262 09BBE D2            C=0    A
263 09BC0 30E           LCHEX  E
264 09BC3 8BD           ?B<=C  A
265 09BC6 A0            GOYES  FLXQ07
266 09BC8 307           LCHEX  7
267 09BCB 5D0           GONC   FLXQ10         (B.E.T.)
268             *
269 09BCE 03    FLXQ05  RTNCC
270             *
271 09BD0 A89   FLXQ07  C=B    P
272 09BD3 81E           CSRB
273 09BD6 A0E           C=C-1  P
274 09BD9 816   FLXQ10  CSRC
275 09BDC 07            C=RSTK
276 09BDE 108           R0=C                  Save a level (only have 3)
277 09BE1 8E00          GOSUBL =FILEP!
          00
278 09BE7 118           C=R0
279 09BEA 06            RSTK=C                Restore level
280 09BEC AC3           D=0    S
281 09BEF 500           RTNNC
282             *
283             * String expression contains legal file name
284 09BF2 137           CD1EX
285 09BF5 135           D1=C
```

```
286 09BF8 8A7          ?C#D   A              NOT AT END OF STRING?
287 09BFB 90           GOYES  FLXQ12
288 09BFD 6CA0         GOTO   FLXQ25
289              *
290 09C01 AF0   FLXQ11 A=0    W
291 09C04 100   FLXQ12 R0=A                  Save file name
292 09C07 14B          A=DAT1 B
293 09C0A 171          D1=D1+ 2              Step over char
294 09C0D 31A3         LCASC  \:\
295 09C11 962          ?A=C   B
296 09C14 21           GOYES  FLXQ13          : ?
297 09C16 7962         GOSUB  BLKOK2         File name followed by char
298 09C1A 500          RTNNC                 other than blank or colon
299              *                            (could be 10 character name)
300 09C1D DB           C=D    A
301 09C1F 135          D1=C                  Position D1 past string
302 09C22 6480         GOTO   FLXQ22
303              *
304 09C26 8E00  FLXQ13 GOSUBL =CNVWUC        Convert to upper case
          00
305              *
306 09C2C 177          D1=D1+ 8
307 09C2F 137          CD1EX
308 09C32 135          D1=C
309 09C35 8BB          ?C<=D  A              At least 5 chars on stack?
310 09C38 50           GOYES  FLXQOK
311 09C3A 582          GONC   flxqpl         (B.E.T.) Device name <4 chars
312              *
313 09C3D D5    FLXQOK B=C    A              Save stk ptr past chars
314 09C3F AF6          C=A    W
315 09C42 3705         LCASC  \TROP\
          F425
          45
316 09C4C 976          ?A#C   W              Not PORT?
317 09C4F 60           GOYES  FLXQ14
318 09C51 6841         GOTO   PRT#P          Parse port # on stack
319              *
320 09C55 D9    FLXQ14 C=B    A              Restore stack ptr past chars
321 09C57 AF8          B=A    W              Copy device name
322 09C5A 8A3          ?C=D   A
323 09C5D 90           GOYES  FLXQ15
324              * More than 4 characters following the colon
325 09C5F 7D12         GOSUB  BLKOK
326 09C63 565   flxqpl GONC   FLXQPL         Not a blank?
327              *
328 09C66 DB    FLXQ15 C=D    A
329 09C68 135          D1=C                  Position D1 past string
330 09C6B D3           D=0    A
331 09C6D 37D4         LCASC  \NIAM\
          1494
          E4
332 09C77 971          ?B=C   W
333 09C7A A2           GOYES  FLXQ19          :MAIN?
334              *
335 09C7C 3734         LCASC  \DRAC\
```

```
            1425
            44
    336 09C86 971          ?B=C   W              :CARD?
    337 09C89 01           GOYES  FLXQ17
    338             *
    339 09C8B 3305         LCASC  \CP\
            34
    340 09C91 975          ?B#C   W              NOT :PCRD EITHER?
    341 09C94 62           GOYES  FLXQPL         4 char device name not recognized
    342             *
    343 09C96 A6F   FLXQ16 D=D-1  B              :PCRD
    344 09C99 2F    FLXQ17 P=     15             :CARD
    345 09C9B 306          LCHEX  6
    346 09C9E AC7          D=C    S
    347 09CA1 B47   FLXQ18 D=D+1  S              :PORT
    348 09CA4 B47   FLXQ19 D=D+1  S              :MAIN
    349 09CA7 110   FLXQ22 A=R0                  RESTORE FILE NAME
    350 09CAA 20    FLXQ25 P=     0              RE-SET P
    351 09CAC A4F          D=D-1  S
    352             * If called DVCTYP, must pop level off stack
    353 09CAF 877          ?ST=1  7
    354 09CB2 00           RTNYES
    355 09CB4 07           C=RSTK
    356 09CB6 02           RTNSC
    357             *
    358 09CB8 07    FLXQ29 C=RSTK                Pop DVCTYP level off stack
    359 09CBA 7597  FLXQPL GOSUB  poll
    360 09CBE 30           CON(2) =pFILXQ
    361 09CC0 470          GOC    FLXQ30         Error ?
    362 09CC3 831          ?XM=0
    363 09CC6 00           RTNYES                Handled successfully ?
    364 09CC8 03    FLXQ30 RTNCC                 Filespec not recognized
    365             *
    366             * FILE NAME IN PROGRAM MEMORY
    367             *
    368 09CCA 136   FILEXQ CDOEX
    369 09CCD 135          D1=C                  PC IN D1
    370 09CD0 07           C=RSTK                Pop RETURN stack from DVCTYP
    371             * ASSUMING A WILL BE BLANK-PADDED
    372 09CD2 8E00         GOSUBL =FILEP+        1ST CHAR MUST BE LEGAL
            00
    373 09CD8 100          R0=A                  Save file name
    374 09CDB B46          C=C+1  S
    375 09CDE 5B0          GONC   FLXQ40         LESS THAN 8 CHARACTERS?
    376             * FOUND  8 CHARACTERS
    377 09CE1 8E00         GOSUBL =CATC++
            00
    378 09CE7 40E          GOC    FLXQ30         FOUND 9TH CHARACTER
    379             *
    380 09CEA 137   FLXQ40 CD1EX
    381 09CED 134          D0=C
    382             *   SEE IF tMAIN, tCARD, OR tPORT FOLLOWS FILE NAME
    383 09CF0 7170         GOSUB  DVCTYP
    384             * Since did return, compensate for subroutine level
    385 09CF4 06           RSTK=C
```

```
386 09CF6 50B          GONC   FLXQ22          (B.E.T.)
387              *
388 09CF9 181   FLXQ65 D0=D0- 2
389 09CFC 35FE         LC(6)  =tPCRD
          10E3
390 09D04 25           P=     5
391 09D06 916          ?A#C   WP
392 09D09 E9           GOYES  FLXQ22
393 09D0B 165          D0=D0+ 6               Step over tPCRD
394 09D0E 578          GONC   FLXQ16         (B.E.T.)
395              *
396 09D11 14A   FLXQ70 A=DAT0 B
397 09D14 3182         LCASC  \(\
398 09D18 962          ?A=C   B
399 09D1B 90           GOYES  FLXQ75
400 09D1D A6F          D=D-1  B               Set D(B) = FF
401 09D20 608F  FLXQ74 GOTO   FLXQ18
402              *
403              *
404              * SAVE FILE NAME & C(S) BEFORE CALLING EXPEXC
405 09D24 110   FLXQ75 A=R0
406 09D27 1F18         D1=(5) =S-R1-0
          8F2
407 09D2E 1517         DAT1=A W               Save file name
408 09D32 07           C=RSTK                 Take unneeded level off stack
409 09D34 161          D0=D0+ 2               Step over (
410 09D37 8E00         GOSUBL =EXPEX+
          00
411 09D3D 7361         GOSUB  PDEV1
412 09D41 17F          D1=D1+ 16              Position past expression on stk
413 09D44 137          CD1EX                  Save D1 in C(A)
414 09D47 1F18         D1=(5) =S-R1-0
          8F2
415 09D4E 1537         A=DAT1 W
416 09D52 100          R0=A                   Restore file name
417 09D55 135          D1=C                   Restore D1
418 09D58 78E1         GOSUB  ave=d1          Update AVMEME
419 09D5C 06           RSTK=C                 Anticipate pop to come
420 09D5E AC3   FLXQ80 D=0    S
421 09D61 6F3F         GOTO   FLXQ18
422              *
423 09D65 15A5  DVCTYP A=DAT0 6
424 09D69 AF3          D=0    W
425 09D6C 161          D0=D0+ 2               Step over 2 nibs
426 09D6F 8E00         GOSUBL =FINDA
          00
427 09D75 00           CON(2) =tCARD
428 09D77 22F          REL(3) FLXQ17
429 09D7A 00           CON(2) =tPORT
430 09D7C 59F          REL(3) FLXQ70
431 09D7F 00           CON(2) =tMAIN
432 09D81 32F          REL(3) FLXQ19
433 09D84 00           CON(2) =tCOLON
434 09D86 23F          REL(3) FLXQ29
435 09D89 00           CON(2) =tXWORD
```

```
436 09D8B E6F          REL(3) FLXQ65
437 09D8E 00           CON(2) =tLITRL
438 09D90 A3F          REL(3) FILEXQ
439 09D93 00           NIBHEX 00
440             *
441 09D95 181          D0=D0- 2
442 09D98 01           RTN
443             *
444             *
445             *
446             ****************************************************************
447             ****************************************************************
448             **
449             ** Name:    PRT#P   -  port# parse
450             **
451             ** Category:   LOCAL
452             **
453             ** Purpose:
454             **      Interprets PORT designation on the expression stack
455             **
456             ** Entry:
457             **      P=0
458             **      Reversed string expression on the stack
459             **      D1 points past ':PORT' at possible port specifier
460             **      D(A) is past last character on stack
461             **
462             ** Exit:
463             **       Via FLXQ18
464             **       P  =  0
465             **       D1 is past the last character on stack
466             **       D(A) contains port# (FFFFF if none specified)
467             **
468             ** Calls:      DIGCK, RANGE, DECHEX, BLKOK2, EOS
469             **
470             ** Uses.......
471             **  Inclusive: A-D, D1, S1
472             **
473             ** Stk lvls:   3
474             **
475             **
476             ** Detail:
477             **       ALLOWS '(d.)' - EG HANGING DECIMAL POINT
478             **       This is not straight-line code ONLY to keep
479             **       GOYES/GOC instructions in range in FILXQ routine
480             **
481             ** History:
482             **
483             **    Date     Programmer             Modification
484             **  --------   ----------   ----------------------------------
485             **  06/29/82   S.W.         Added documentation
486             **  02/03/83   S.W.         No digit required before decimal pt
487             **
488             ****************************************************************
489             ****************************************************************
490             *
```

```
491                *
492 09D9A DB     PRT#P  C=D    A
493 09D9C 06            RSTK=C                Save pointer on stack
494 09D9E 137           CD1EX
495 09DA1 135           D1=C
496 09DA4 25            P=     5
497 09DA6 A83           D=0    P
498 09DA9 20            P=     0
499 09DAB E3            D=D-C  A              Length of string
500 09DAD 81F           DSRB                  #CHARS IN STRING
501 09DB0 CF            D=D-1  A
502 09DB2 485           GOC    PRT#NL         CARRY=> NO PORT DESIGNATION
503                *
504 09DB5 14B           A=DAT1 B              READ IN NEXT CHARACTER ON STACK
505 09DB8 3182          LCASC  \(\
506 09DBC 966           ?A#C   B
507 09DBF 54            GOYES  PRT#N+         Extraneous char after PORT
508 09DC1 D1            B=0    A
509 09DC3 77C0          GOSUB  EOS
510 09DC7 463           GOC    FSPCER         At end of string?
511 09DCA 851           ST=1   1              No digit found prior to dec pt
512 09DCD 31E2          LCASC  \.\
513 09DD1 962           ?A=C   B
514 09DD4 51            GOYES  PRT#P5
515 09DD6 841           ST=0   1
516 09DD9 7370          GOSUB  DIGCK+         MUST find digit
517 09DDD 5A3           GONC   PRT#P7         FOUND ')'?
518                * SO FAR FOUND '(d'
519 09DE0 31E2          LCASC  \.\
520 09DE4 966           ?A#C   B
521 09DE7 71            GOYES  FSPCER
522                * the following 2 commands are to allow '(d.)'
523                * if want to cut code, can delete the next 2 commands, iff the
524                * gosub following is changed to 'GOSUB DIGCK'
525 09DE9 7370   PRT#P5 GOSUB  DIGCK2
526 09DED 552           GONC   PRT#P6         '.' FOLLOWED BY ')' ?
527 09DF0 7C50          GOSUB  DIGCK+         NO, LOOK FOR DIGIT
528 09DF4 562           GONC   PRT#P8
529                * NOW HAVE '(d.d'
530 09DF7 7550          GOSUB  DIGCK+
531 09DFB 522           GONC   PRT#P9
532                * FOUND '(d.dd' - SHOULD HAVE FOUND ')'
533 09DFE 8C00 =FSPCER GOLONG =INVFSP        Invalid filespec
          00
534                *
535 09E04 7B70   PRT#N+ GOSUB  BLKOK2
536 09E08 55F    fspcer GONC   FSPCER         Not a blank?
537 09E0B D3     PRT#NL D=0    A
538 09E0D A6F           D=D-1  B              Set port spec. to FF
539 09E10 463           GOC    flxq18         (B.E.T.)
540                *
541 09E13 871    PRT#P6 ?ST=1  1              No digits found yet?
542 09E16 8E            GOYES  FSPCER
543 09E18 BB1    PRT#P7 BSL    X              Move port# to B(XS)
544 09E1B BB1    PRT#P8 BSL    X              B(B) contains port extender
```

```
545                *
546 09E1E 3251  PRT#P9 LCHEX  615
          6
547 09E23 9E1          ?B>C   B              PORT EXTENDER# > 15?
548 09E26 8D           GOYES  FSPCER
549 09E28 9A9          ?B>=C  XS             PORT# > 5?
550 09E2B 3D           GOYES  FSPCER
551 09E2D AF0          A=0    W
552 09E30 AE4          A=B    B              PORT EXTENDER#
553 09E33 BB5          BSR    X              Get port# in nib 1
554 09E36 D9           C=B    A              Copy nib 1 (port#) & 2-4 (zeroes)
555 09E38 D7           D=C    A              SAVE PORT#
556 09E3A 8F00         GOSBVL =DECHEX
          000
557                * UPON RTN FROM DECHEX C=0
558 09E41 A86          C=A    P
559 09E44 A87          D=C    P
560 09E47 07    flxq18 C=RSTK
561 09E49 135          D1=C                  Position D1 past string
562 09E4C 611F         GOTO   FLXQ80         Sets D(S)=0; on to FLXQ18
563                *
564                * CARRY CLR=> FOUND DIGIT, ')' AND END OF STRING
565                *       SET=> FOUND DIGIT AND A(B) CONTAINS NEXT CHARACTER
566                * Error exits if digit not found, or if ')' is found but not at
567                *  the end of the string
568                *
569 09E50 8F00  DIGCK+ GOSBVL =DRANGE
          000
570 09E57 46A          GOC    FSPCER         NOT A DIGIT?
571 09E5A BB1          BSL    X
572 09E5D A88          B=A    P
573 09E60 7A20  DIGCK2 GOSUB  EOS
574 09E64 499          GOC    FSPCER         AT END OF STRING?
575 09E67 3192         LCASC  \)\
576 09E6B 966          ?A#C   B
577 09E6E 00           RTNYES
578                *
579 09E70 7A10         GOSUB  EOS
580 09E74 490          GOC    DIGCK3         Blank found?
581 09E77 7800         GOSUB  BLKOK2
582 09E7B 5C8          GONC   fspcer
583 09E7E 03    DIGCK3 RTNCC
584                *
585 09E80 14B  =BLKOK  A=DAT1 B
586 09E83 3102  BLKOK2 LCASC  \ \
587 09E87 962          ?A=C   B
588 09E8A 00           RTNYES
589 09E8C 01           RTN
590                *
591 09E8E CF    EOS    D=D-1  A              AT END OF STRING?
592 09E90 400          RTNC
593 09E93 171          D1=D1+ 2
594 09E96 14B          A=DAT1 B
595 09E99 01           RTN
```

```
596                     STITLE Port Address Evaluation (PDEV)
597             ********************************************************************
598             ********************************************************************
599             **
600             ** Name:(S) PDEV    -  Evaluate Num Expression as Port Device
601             ** Name:    PDEV+   -  Evaluate Num Expression as Port Device
602             ** Name:    PDEV1   -  Evaluate Num Expression as Port Device
603             **
604             ** Category:   FILUTL
605             **
606             ** Purpose:  Evaluates numeric expression for port address
607             **
608             **           PDEV+ and PDEV entries evaluate an expression
609             **           in memory and ensure it is a valid numeric
610             **           expression.
611             **
612             **           PDEV1 assumes that the evaluated expression is
613             **           already on the stack.  It is useful for functions.
614             **
615             ** Entry:    3 entry points:
616             **           1) PDEV+ - D0 2 nibs prior to alleged numeric
617             **                      expression.
618             **           2) PDEV  - D0 at alleged numeric expression.
619             **           3) PDEV1 - D1 points to evaluated expression on
620             **                      math stack.
621             **
622             ** Exit:     D(0)=Port extender#; D(1)=port#
623             **           D1 points to numeric expression on stack
624             **           D0 past evaluated numeric expression
625             **            (if entered at PDEV1, D0 unchanged from entry)
626             **           Statuses intact
627             **            (except if entered at PDEV1)
628             **           ERROR EXITS IF EVALUATED EXPRESSION IS INAPPROPRIATE
629             **           FOR A PORT ADDRESS.
630             **
631             ** Calls:    EXPEXC, TST12A, FRAC15, FLTDH, ARGSTA
632             **           CLRFRC, GTPRT#, RSTST
633             **
634             ** Uses:     A-D, D1,D0, R0-R3, all of function scratch -- EXPEXC
635             **
636             ** Detail:   Allows numeric expressions which evaluate to x.yy
637             **           where:
638             **           0<= x <= 5    and   0<= yy <= 15
639             **
640             ** Stack lvls: 5
641             **
642             ** History:
643             **
644             **    Date     Programmer   Modification
645             **  --------   ----------   ------------
646             **  06/29/82   S.W.         Added documentation
647             **  08/05/82   S.W.         Added PDEV1 entry point
648             **
649             ********************************************************************
650             ********************************************************************
```

```
651                *
652                *
653                *
654 09E9B 161   =PDEV+  D0=D0+ 2              STEP OVER tPORT
655 09E9E 8E00 =PDEV   GOSUBL =EXPEX+         Save statuses
           00
656 09EA4 8E00 =PDEV1  GOSUBL =ARGSTA         Errors on complex,NaN,array
           00
657 09EAA 470          GOC    CHKSGN          Infinite?
658 09EAD 958          ?A=0   M               Zero?
659 09EB0 70           GOYES  PDEV3           Want to allow negative zero
660 09EB2 94C   CHKSGN ?A#0   S               Negative?
661 09EB5 66           GOYES  PDVE24
662 09EB7 2C    PDEV3  P=     12
663 09EB9 AF2          C=0    W               Exponent = 000
664 09EBC 3251         LCHEX  515             load up 5.15
           5
665 09EC1 23           P=     3               Set for TST12A
666 09EC3 8E00         GOSUBL =TST12A         TESTS <=5.15 (TRAPS OUT INF.)
           00
667 09EC9 515          GONC   PDVE24
668                * 0 <= X <= 5.15
669 09ECC E4           A=A+1  A               Increment exponent by 2
670 09ECE E4           A=A+1  A                to move dec. pt past hundredths
671 09ED0 8E00         GOSUBL =CLRFRC         TRUNCATE THOUSANDTHS
           00
672 09ED6 CC           A=A-1  A               Restore original exponent
673 09ED8 CC           A=A-1  A
674                * In A now have original expression, with thousandths truncated
675 09EDA 100          R0=A                   SAVE EXPONENT
676 09EDD AF9          C=B    W
677 09EE0 109          R1=C                   SAVE MANTISSA
678 09EE3 8E00         GOSUBL =CLRFRC         truncate (no return 12-dig form)
           00
679 09EE9 D3           D=0    A               In B now have port#
680 09EEB 7A10         GOSUB  GTPRT#          Convert to hex
681 09EEF 05           SETDEC                 Port# in D(0)
682                * PORT# IN D(0)
683 09EF1 BE3          DSL    B
684 09EF4 110          A=R0                   RESTORE EXPONENT
685 09EF7 119          C=R1
686 09EFA AF5          B=C    W               RESTORE MANTISSA
687 09EFD 8E00         GOSUBL =FRAC15         Port extender# in B
           00
688 09F03 B34          A=A+1  X
689 09F06 B34          A=A+1  X               Make it an integer
690                *
691 09F09 AD4   GTPRT# A=B    M               GET MANT FROM 15-DIG FORM
692 09F0C 8E00         GOSUBL =fltdh          Convert from floating dec
           00
693 09F12 31F0         LCHEX  0F               to hex
694 09F16 9EA          ?A<=C  B               Ensure <= 15 (for port extender#)
695 09F19 80           GOYES  GTPR#1
696                *
697 09F1B 8C00  PDVE24 GOLONG =ARGERR         Invalid argument
```

```
            00
    698             *
    699 09F21 A86   GTPR#1 C=A    P
    700 09F24 A87          D=C    P
    701 09F27 8C00  rstst  GOLONG =RSTST        Restore statuses
            00
    702             *
    703             *
```

```
                STITLE File Spec Execute
*****************************************************************
*****************************************************************
**
** Name:(S) FSPECx  -  File Specification Execute
**
** Category:   FILUTL
**
** Purpose:  Evaluates a file specification
**
** Entry:    DO @ File specification start
**
** Exit :    DO past file specification
**           Carry Clear: Legal file specification
**
**              A = filename  (blank filled)
**              A = 0 if no filename
**              RO = last two chars of file name (if any)
**                 = two blanks by default
**              D(S) = F  NO DEVICE SPECIFIED
**                     0  MAIN
**                     1  PORT    D(B) = PORT number
**                                D(B) = FF if :PORT
**                     7  CARD    D(B) = 0 if CARD
**                        PCRD    D(B) # 0 if PCRD
**                   = 8  HP-IL device  (D(X) = device addr)
**                   > 8   other
**
**              P=0
**              P reset before POLL
**              If file specifier was a string expression:
**               (AVMEME) points past the string on the stack
**
**
**           Carry set:
**              Unrecognized File Specification
**              C(3-0) = Error#
**
** Calls:    FILXQ^, POLL
**
**           FILXQ^ returns:
**           Carry Clear ---> Illegal File Spec
**                            S-R1-0 holds original DO
**           Carry Set   ---> Legal File Spec
**                            S8=0  Simple Filename
**                            S8=1  D(S)=F No Device specified
**                                        0 MAIN
**                                        1 PORT
**                                          D(B) = Port#
**                                               = FF if :PORT
**                                        7 CARD
**                                          PCRD
**                                          D(B) = 0 if CARD
**                                               # 0 if PCRD
**
```

```
759             ** Uses:     A-D
760             **           D = End of Expression stack  (from FILXQ^)
761             **           STMTDO, STMTR1 (all of it), S1,S2,S7 -- FILXQ
762             **           D1,D0, R0-R3, all of function scratch -- EXPEXC
763             **
764             ** Detail: Try Mainframe File Execute                (FILXQ^)
765             **         Blank-fill lower 2 bytes of R0
766             **         If acceptable file specification  (Carry set)
767             **            If simple filename
768             **               Set Device = 0   (D(S))
769             **            RTNCC
770             **         else
771             **            POLL for File Specification Execute
772             **            Return if Carry Set
773             **            If handled (XM=0)
774             **               Return with Carry Clear
775             **            else
776             **               C <-- eFSPEC
777             **               Return with Carry Set
778             **
779             ** Stack lvls: 6
780             **
781             ** History:
782             **
783             **    Date     Programmer   Modification
784             **  --------   ----------   ------------
785             **  06/29/82   S.W.         Added documentation
786             **
787             ********************************************************************
788             ********************************************************************
789             *
790             *
791             ********************************************************************
792             ********************************************************************
793             **
794             ** Name:(S) pFSPCx  -  File Spec Execution poll
795             **
796             ** Category:   POLL
797             **
798             ** Type:       POLL
799             **
800             ** Purpose:
801             **      POLL for file specification execution
802             **
803             ** Should poll be "Handled" (return with XM=0)?:
804             **      Yes
805             **
806             ** Meaning of "Handling" Poll (what does code do if handled?):
807             **      Returns 1st 8 chars of file name in A
808             **       and last two characters in R0
809             **      D(S)=Device type; D(X)= Device address
810             **
811             ** Entry conditions for handler (registers, ST, RAM, etc.):
812             **      Carry clear
813             **      B[A] = Poll number.
```

```
814             **      HEX mode.
815             **      P=0.
816             **      Low 2 bytes of R0 are blank-filled
817             **      S7=1 => String expression on stack
818             **              Top of stack pointed to by AVMEME
819             **              D0 past string expression
820             **        =0 => Literal
821             **              D0 may be restored from STMTD0 to interpret
822             **               file specifier
823             **
824             ** Normal exit conditions from handler if handled (ST, RAM,
825             ** registers, etc.):
826             **      Carry clear
827             **      HEX mode.
828             **      XM=0.
829             **      A contains file name (blank-filled)
830             **       If > 8 characters, last 9 & 10 in R0
831             **      If no file name specified, A=0
832             **      D(S) = Device type
833             **      D(X) = Device address
834             **      D0 past file specifier
835             **      S7 intact from entry
836             **      If S7 set on entry (string), D1 must point past file
837             **       specifier on stack; eg D1 must reflect new top of stk
838             **
839             ** Normal exit conditions from handler if not handled (ST, RAM,
840             ** registers, etc.):
841             **      Carry clear
842             **      HEX mode.
843             **      XM=1.
844             **
845             ** Error exit conditions from handler:
846             **      Carry set.
847             **      HEX mode.
848             **      C[0-3] = Error number.
849             **
850             ** Available subroutine levels:
851             **      7
852             **
853             ** NOTE:
854             **      If not handled, error generated is eFSPEC
855             **
856             ** What registers/RAM may be used if handled?:
857             **      A-D, D0, D1, P
858             **      R0,R1, S-R1-0 thru S-R1-3, STMTD0, STMTD1,
859             **      S1,S2
860             **
861             ** What registers/RAM may be used if not handled?:
862             **      A-D, D0, D1, P
863             **      Same as if handled (See above)
864             **      S7 must remain intact
865             **
866             ** What registers/RAM may be used if error exit:
867             **      A-D, D0, D1, P
868             **      Same as if handled (See above)
```

```
869                ;;
870                ;; Special memory/pointer considerations (are pointers funny?):
871                ;;      No
872                ;;
873                ;; Envisioned application(s):
874                ;;      PURGE A:TAPE
875                ;;
876                ;; Note:
877                ;;      If not handled, error generated is eFSPEC.
878                ;;
879                ;; History:
880                ;;
881                ;;    Date     Programmer              Modification
882                ;; --------   ----------   ----------------------------------
883                ;;  02/04/83   S.W.         Added documentation
884                ;;
885                ;;;
```

```
                STITLE Find File in RAM ROM PORTS
*****************************************************************
*****************************************************************
**
** Name:(S) FINDF   -  Find a file
** Name:(S) FINDF+  -  Find a file
** Name:    FILEMF  -  Find a file
** Name:(S) FILEF   -  Find a file
** Name:    FINDWF  -  Find a file
**
** Category:   FILUTL
**
** Purpose:  Searches for a Specified File in file chain(s)
**           specified by the caller.
**
**           The entry points which allow the file chains to
**           be specified require as entry conditions some
**           of the exit conditions from FILXQ^/FSPECx.
**
**           FILEF and FILEMF entries search the MAIN file
**           chain only.
**
**           FINDF and FINDF+ entries look at D(S) to
**           determine which file chains to search.  The only
**           difference between the two entry points is that
**           FINDF assumes the integrity of D(S) and A(W),
**           whereas FINDF+ checks their integrity to ensure
**           that A(W) is nonzero and D(S)<=6.
**
**           FINDWF searches the MAIN file chain for <workfile>
**
** Entry: P=0
**        5 entry points:
**
**        1) FINDF -  file name in A(W)
**                    D(S) determines search pattern:
**                       =F => Search MAIN, plug-ins
**                       =0 => Search MAIN only
**                    other => Search Plug-ins only
**                             D(B) indicates port desig.
**                                =FF  =>   all PORTS (:PORT)
**                             OR D(1)=     PORT #
**                                D(0)=     Extender #
**
**        2) FILEF -  File name in A(W) - Mainframe search only
**
**        3) FILEMF   same as above, except file name in B(W)
**
**        4) FINDF+ - Same as FINDF.
**
**        5) FINDWF - Searches for workfile
**
** Exit:     P=0
**           Carry Clear - File found
**             D1   @  File Start
```

```
 972            **            A(W)=B(W) contain file name
 973            **            D(S) =  Device Type
 974            **
 975            **                    0 = Mainframe RAM
 976            **                    1 = IRAM
 977            **                    2 = ROM
 978            **                    3 = EEPROM
 979            **
 980            **               It cannot be assumed that Device Type is
 981            **               limited to these numbers.
 982            **
 983            **               Routines using FINDF should probably POLL when
 984            **               Device Type is not 0-2.
 985            **
 986            **            D(B) =  ExtenderW, Port#   (if applicable)
 987            **
 988            **          Carry Set => File not found
 989            **            S6=1 =>
 990            **              B=A  = Filename
 991            **              C(3-0) contains errW for eFnFND or eDVCNF
 992            **            S6=0 (FINDF+ entry only) =>
 993            **              Illegal file spec for file chain search
 994            **               either A(W)=0 or D(S)>=7
 995            **              C(3-0)=eFSPEC
 996            **              C(S) = 2*(D(S)+1)
 997            **
 998            ** Calls:    ROMCHK, ROMFND, ROMF-1, FILSKP, C=MAIN, WRKFIL
 999            **
1000            ** Uses:     A-D, D1, S6,S8,R1,R2 (if outside of Main search)
1001            **                              R3 (if single PORT search)
1002            **
1003            **          S6 = Not Initial PORT search
1004            **          S8 = Single/ Special file chain search
1005            **
1006            **          ROMCHK
1007            **          ROMFND  uses A-D,D1,R0,R1
1008            **          ROMF-1  uses A-D,D1,R0,R1,R3
1009            **
1010            ** Stk lvls: 2
1011            **
1012            ** Detail:
1013            **
1014            **  FINDF+:  If D(S) >= 7 or A(W)=0
1015            **             Return with carry set; C(3-0)=eFSPEC
1016            **  FINDF:   Clear Single Filechain Search flag  (S8)
1017            **           Move filename to B
1018            **           If Standard search    D(S)=F
1019            **              goto 1;
1020            **           If MAINframe only     D(S)=0
1021            **              goto FILFMF;
1022            **           else                  (PORT)
1023            **              Save filename      (R2)
1024            **              If all Ports     (D(B)=FF)
1025            **                 go Search ALL Ports  (goto 3);
1026            **              else
```

```
1027            **                Set single file chain flag (S8)
1028            **                Find Start of file chain in Port  (ROMF-1)
1029            **                Restore filename to A
1030            **                Put filename in B
1031            **                Set S6 for error  (file not found)
1032            **                If not found
1033            **                   Return Carry   C(3-0)=eDVCNF
1034            **                else
1035            **                   Continue search (goto 2)
1036            **   FILEF:    B <-- Filename
1037            **  FILFMF:    Set Single Filechain flag      (S8)
1038            **       1:    Set pointer @ Main memory start
1039            **             Clear Initial Port Search flag (S6)
1040            **       2:    Read filename
1041            **             If not at end of file chain    (A(B)#0)
1042            **                If filename match --> RTNCC
1043            **                else
1044            **                   Skip to next file
1045            **                   goto 2;
1046            **             else  (End of file chain)
1047            **                Restore file name to A
1048            **                If single search only          (S8)
1049            **                   RTNC   C(3-0)=eFnFND
1050            **                else
1051            **                   If initial PORT search     (S6=0)
1052            **                      Save filename           (R2)
1053            **                      Set Not Initial PORT search  (S6)
1054            **     3:               If no ROMs exist        (ROMCHK)
1055            **                         Restore filename
1056            **                         RTNC   C(3-0)=eFnFND
1057            **                      else
1058            **                         Set B = filename
1059            **                         go search ROM for file   (goto 2)
1060            **                   else
1061            **                      Find next ROM               (ROMFND)
1062            **                      Restore filename
1063            **                      If no more ROMs  ---> RTNC C(3-0)=eFnFND
1064            **                      Set B = filename
1065            **                      go Search ROM for file      (goto 2);
1066            **
1067            ** Note:
1068            **      Device ID's 2-6 are NOT available for use.
1069            **      Dedicated devices are restricted to ID's 9-E
1070            **
1071            ** History:
1072            **
1073            **    Date      Programmer   Modification
1074            **  --------    ----------   ------------
1075            **  06/29/82    S.W.         Added Documentation
1076            **  10/29/82    S.W.         Modified entry conditions for
1077            **                            new device codes
1078            **  12/20/82    S.W.         Calls FILSKP instead of RDHDR
1079            **                            so FINDF doesn't use S9
1080            **
1081            ********************************************************************
```

```
1082                ************************************************************************
1083                *
1084 09F63 846  =FINDF+ ST=0   6              If rtn w/carry set & S6=0 =>
1085 09F66 ACB          C=D    S              D(S) >= 7 => Illegal filespec
1086 09F69 B46          C=C+1  S
1087 09F6C A46          C=C+C  S
1088 09F6F 4BE          GOC    FSPC30
1089 09F72 978          ?A=0   W              No filename => Illegal filespec
1090 09F75 6E           GOYES  FSPC30
1091                *
1092                * Move filename to B
1093                * Clear Single Filechain flag
1094                * If no device
1095                *    go Search Main, ROM
1096                *
1097 09F77 AF8  =FINDF  B=A    W              Save filename
1098 09F7A 848          ST=0   8              Clear Single Filechain flag
1099 09F7D B47          D=D+1  S
1100 09F80 453          GOC    FILF05         NO DEVICE SPECIFIED?
1101                *
1102                * If Device = :MAIN
1103                *    go Search MAINframe ONLY
1104                *
1105 09F83 A4F          D=D-1  S
1106 09F86 A4F          D=D-1  S
1107 09F89 492          GOC    FILFMF         :MAIN?
1108                *
1109                * Device = :PORT
1110                *   Save filename in R2
1111                *   If all Port search  (D(B)=FF)
1112                *      go Search ALL ports
1113                *
1114 09F8C 102          R2=A                  Save filename
1115 09F8F B67          D=D+1  B              Search ALL ports ? D(B)=FF
1116 09F92 4C5          GOC    FILF50
1117                *
1118                * Set Single Filechain Search flag  (S8)
1119                * Find start of file chain for particular port
1120                * Restore filename  (R2 --> A)
1121                * If not found
1122                *    Return Carry
1123                * else
1124                *    Move filename to B
1125                *    go Search file chain for file
1126                *
1127 09F95 858          ST=1   8              Single Filechain flag
1128 09F98 7080         GOSUB  ROMF-1
1129 09F9C 112          A=R2                  Restore filename
1130 09F9F 3300         LC(4)  =eDVCNF
           00
1131 09FA5 6650         GOTO   FILF60
1132                *
1133                * Find workfile - Mainframe search only
1134                *
1135 09FA9 7636 =FINDWF GOSUB  WRKFIL
```

```
1136 09FAD AFA   FINDW+ A=C    W
1137            *
1138            * MAINframe Search ONLY - Entry
1139            *
1140 09FB0 AF8   =FILEF  B=A    W              Move Filename to B
1141 09FB3 858   =FILFMF ST=1   8
1142            *
1143            * Set Start of Search @ Mainframe File Chain Start
1144            * Clear NOT Initial PORT search flag
1145            * Clear Device field to return MAIN frame find
1146            *
1147 09FB6 7650  FILF05 GOSUB  C=MAIN          Set C(A)=MAINST
1148 09FBA AC3          D=0    S               Clear Device field
1149 09FBD 846          ST=0   6               Clear Not Initial PORT search
1150            *
1151            * Search File Chain for Match
1152            *
1153 09FC0 135   FILF10 D1=C
1154 09FC3 1537  FILEF- A=DAT1 W               Read Filename
1155 09FC7 968          ?A=0   B               End of File Chain ?
1156 09FCA 51           GOYES  FILF40
1157 09FCC 974          ?B#A   W               NOT A MATCH?
1158 09FCF 40           GOYES  FILF30
1159            *
1160            * Match found ---> RTNCC
1161            *
1162 09FD1 03           RTNCC                  WANT CARRY CLR FOR MATCH
1163            *
1164            * Skip to next file
1165            *
1166 09FD3 133   FILF30 AD1EX                  A points to file header
1167 09FD6 8E00         GOSUBL =FILSK+
           00
1168 09FDC 53E          GONC   FILF10          (B.E.T.) C(A) @ Next file
1169            *
1170            * End of File Chain
1171            * Restore filename to A
1172            * If Single Filechain search --> RTNC
1173            *    File not found
1174            *
1175 09FDF       FILF40
1176 09FDF AF4          A=B    W               Restore filename
1177 09FE2 878          ?ST=1  8               Single file chain search?
1178 09FE5 11           GOYES  FILF55          CARRY -> File not found
1179            *
1180            * If Initial PORT search
1181            *    Save filename
1182            *    Restore filename to A
1183            *    If no ROMs exist
1184            *       RTNC
1185            *    Set not Initial PORT search  (S6)
1186            *    go Search ROM for file
1187            *
1188 09FE7 876          ?ST=1  6               Not initial PORT search ?
1189 09FEA E1           GOYES  FILF70
```

```
1190 09FEC 102          R2=A
1191            *
1192            * Entry for search ALL ports
1193            *    Filename already saved in R2
1194            *    Set NOT initial PORT search
1195            *
1196 09FEF 7454  FILF50 GOSUB  romchk        Check if any ROMs
1197 09FF3 112   FILF52 A=R2                 Restore filename
1198 09FF6 3300  FILF55 LC(4)  =eFnFND
          00
1199 09FFC AF8   FILF60 B=A    W
1200 09FFF 856          ST=1   6
1201 0A002 400          RTNC                No ROMS - File not found
1202 0A005 5DB          GONC   FILEF-       (B.E.T.)
1203            *
1204            * Find Next ROM
1205            * Restore filename
1206            * If no more ROMs ---> RTNC
1207            * else
1208            *    go Search ROM for file
1209            *
1210 0A008 7144  FILF70 GOSUB  romfnd        Find next ROM
1211 0A00C 66EF         GOTO   FILF52
1212            *
1213 0A010 1F85 =C=MAIN D1=(5) =MAINST
          5F2
1214 0A017 147          C=DAT1 A
1215 0A01A 01           RTN
1216            *
1217            *
```

```
1218                    STITLE Find a particular ROM
1219                ****************************************************************
1220                ****************************************************************
1221                **
1222                ** Name:    ROMF-   -  Find file Chain start for a ROM
1223                ** Name:    ROMF-1  -  Find file Chain start for a ROM
1224                **
1225                ** Category: GENUTL
1226                **
1227                ** Purpose:  Find file chain start for a specific ROM
1228                **
1229                ** Entry:
1230                ** 2 entry points:
1231                **       1) ROMF-  -  D(0)=Extender#; D(1)=Port#
1232                **       2) ROMF-1 -  Same as above except D(B)=D(B)+1
1233                **
1234                ** Exit:     Carry Clear - PORT found
1235                **             D1 @ Filechain start
1236                **             A  = Filename of first file in chain
1237                **            D(S)= Device type
1238                **            R3  = D(B) on entry
1239                **
1240                **          Carry Set - PORT not found
1241                **
1242                ** Calls:    ROMCHK, ROMFND
1243                **
1244                ** Uses:
1245                **  Inclusive:  A-D,R1,R3,D1
1246                **
1247                **           ROMCHK
1248                **           ROMFND uses  A-D,D1,R1
1249                **
1250                ** Stk lvls: 1
1251                **
1252                ** Detail:   Save D --> R3
1253                **           If no ROMS    (ROMCHK)
1254                **              RTNC
1255                **           else
1256                **              If Extender# and Port# match
1257                **                 RTNYES
1258                **              else
1259                **           Continue until no more ROMs  (RTNC)
1260                **
1261                ** History:
1262                **
1263                **    Date     Programmer   Modifications
1264                **  --------   ----------   -------------
1265                **  08/09/82   S.W.         Updated documentation & added
1266                **                           ROMF- entry point
1267                **
1268                ****************************************************************
1269                ****************************************************************
1270 0A01C A6F    =ROMF-1 D=D-1  B              Assumes D(B) was incremented to ck
1271 0A01F DB     =ROMF-  C=D    A              for no ports specified - Save D(B)
1272 0A021 10B           R3=C
```

```
1273 0A024 7F14         GOSUB  ronchk        Look for ROMs
1274 0A028 400          RTNC                 No ROMs found
1275 0A02B 590          GONC   ROMF-3
1276 0A02E 7B14  ROMF-2 GOSUB  ronfnd        Find next ROM
1277 0A032 400          RTNC                 No more ROMs
1278 0A035 11B   ROMF-3 C=R3                 Restore original Port#
1279 0A038 967          ?C#D   B             no Match ?
1280 0A03B 3F           GOYES  ROMF-2        Continue
1281 0A03D 03           RTNCC                Port found
```

```
1282                    STITLE PURGE Execute
1283            ********************************************************************
1284            ********************************************************************
1285            **
1286            ** Name:    PURGE   -  PURGE statement execute
1287            **
1288            ** Category:   STEXEC
1289            **
1290            ** Purpose:  Executes PURGE statement
1291            **
1292            ** Entry:    DO POINTS PAST tPURGE
1293            **
1294            ** Exit:     Either go to NXTSTM
1295            **              or  ERROR exit if
1296            **                          specified file is SECURE
1297            **
1298            ** Calls:    FINDF, PRGFMF, tKYSck, GETPRO, POLL, MFWRQ8,
1299            **           FSPECx, CURDVC, EOLXCK, FILSKP, PRGFMF, C=MAIN
1300            **
1301            ** Uses:     A-D, S-R0-0, R0&R1 (RFADJ-), R2, D1,D0
1302            **           S-R0-1 (PRGFMF)
1303            **           R0-R3, STMTD0, STMTR1 (All of it), S1,S2,S7,
1304            **            + all of function scratch  -- FSPECx
1305            **
1306            ** Stack lvls: 7
1307            **
1308            ** Detail:     Does not error exit if file not found -
1309            **             issues warning
1310            **
1311            **             PURGE of current running file collapses ALL
1312            **             environments.
1313            **
1314            ** History:
1315            **
1316            **    Date     Programmer   Modification
1317            ** ---------   ----------   ------------
1318            **  06/29/82   S.W.         Added documentation.
1319            **  07/29/82   S.W.         Warning issued if specified
1320            **                           file doesn't exist.
1321            **  11/24/82   S.W.         Collapse of current running
1322            **                           program collapses ALL
1323            **                           environments.
1324            **
1325            ********************************************************************
1326            ********************************************************************
1327            *
1328            ********************************************************************
1329            ********************************************************************
1330            **
1331            ** Name:(S) pPURGE  -  Poll to PURGE file on external device
1332            **
1333            ** Category:   POLL
1334            **
1335            ** Type:       POLL
1336            **
```

```
1337            ** Purpose:
1338            **      Polls to PURGE a file on non-mainframe device
1339            **
1340            ** Should poll be "Handled" (return with XM=0)?:
1341            **      Yes
1342            **
1343            ** Meaning of "Handling" Poll (what does code do if handled?):
1344            **      Purges the file
1345            **      The mainframe will handle purging any associated FIB
1346            **
1347            ** Entry conditions for handler (registers, ST, RAM, etc.):
1348            **      Carry clear
1349            **      B[A] = Poll number.
1350            **      HEX mode.
1351            **      P=0.
1352            **      D0 past file specifier
1353            **      D(S) & D(X) contains device info
1354            **      Blank-filled file name in A(W) & R0; R0 contains chars
1355            **       9 & 10; If no file name given, A=0
1356            **
1357            ** Normal exit conditions from handler if handled (ST, RAM,
1358            ** registers, etc.):
1359            **      Carry clear
1360            **      HEX mode.
1361            **      XM=0.
1362            **      S8=0 (indicates current file was not purged)
1363            **
1364            ** Normal exit conditions from handler if not handled (ST, RAM,
1365            ** registers, etc.):
1366            **      Carry clear
1367            **      HEX mode.
1368            **      XM=1.
1369            **
1370            ** Error exit conditions from handler (POLL only):
1371            **      Carry set.
1372            **      HEX mode.
1373            **      C[0-3] = Error number.
1374            **
1375            ** Available subroutine levels:
1376            **      7
1377            **
1378            ** NOTE:
1379            **      If not handled, error generated is eFSPEC
1380            **
1381            **      --SCRATCH RAM TO CONSIDER BELOW:--
1382            **      --STMT/FN Scratch, SCRTCH, SNAPBF, TRFMBF, LDCSPC,--
1383            **      --LEXPTR.--
1384            **
1385            ** What registers/RAM may be used if handled?:
1386            **      A-D, D0, D1, P
1387            **      Anything available to statements
1388            **      STMT/FN scratch, R0-R4, S0-S11
1389            **
1390            ** What registers/RAM may be used if not handled?:
1391            **      A-D, D0, D1, P
```

```
1392                **      Same as if handled, except don't use RO !
1393                **
1394                ** What registers/RAM may be used if error exit:
1395                **      A-D, D0, D1, P
1396                **      Same as if handled
1397                **
1398                ** Special memory/pointer considerations (are pointers funny?):
1399                **      No
1400                **
1401                ** Envisioned application(s):
1402                **      PURGE A:TAPE
1403                **      PURGE :PORT(2)  ! PURGE ALL on a plug-in EPROM perhaps
1404                **
1405                ** Note:
1406                **      If not handled, error generated is eFSPEC.
1407                **
1408                ** History:
1409                **
1410                **    Date      Programmer              Modification
1411                **  --------   ----------   ----------------------------------
1412                **  06/29/82   S.W.         Added documentation
1413                **
1414                ********************************************************************
1415                ********************************************************************
1416                *
1417                *
1418 0A03F 00000          REL(5) =PURGDC
           0
1419 0A044 00000          REL(5) =PURGEP
           0
1420 0A049 74B0 =PURGE    GOSUB  eolxc+
1421 0A04D 5E1            GONC   PURG20          NOT AT EOL?
1422                * PURGE CURRENT FILE
1423 0A050 77B5           GOSUB  CURDVC          GET DEVICE FOR CURR FILE
1424 0A054 7EE0  PURG00   GOSUB  PRGFMF          ATTEMPT TO PURGE THE FILE
1425 0A058 404            GOC    bserr           ERROR?
1426 0A05B 877   PURG10   ?ST=1  7               PURGED CURRENT RUNNING FILE?
1427 0A05E 80             GOYES  TRMPRG          TERMINATING PURGE
1428 0A060        PURGNF
1429 0A060 8C00   PURGDN   GOLONG =NXTSTM
           00
1430                *
1431 0A066 8C00   TRMPRG   GOLONG =ENDALL
           00
1432                *
1433 0A06C 7DE2   PURG20   GOSUB  tKYSck
1434 0A070 521            GONC   PURG25          tKEYS?
1435 0A073 3100           LC(2)  =tALL
1436 0A077 962            ?A=C   B
1437 0A07A E3             GOYES  PURG40
1438                *
1439 0A07C 7DAE           GOSUB  FSPECx
1440 0A080 481            GOC    bserr           ERR# LOADED IF CARRY
1441                *
1442 0A083 7CDE   PURG25   GOSUB  FINDF+
```

```
1443 0A087 5CC          GONC   PURG00          FILE FOUND?
1444            * GIVE WARNING THAT FILE DOESN'T EXIST
1445 0A08A 866          ?ST=0  6               Illegal file spec?
1446 0A08D 21           GOYES  PRGEXT
1447 0A08F 8E00         GOSUBL =MFWRQ8
           00
1448 0A095 6ACF         GOTO   PURGNF
1449            *
1450 0A099 8C00  bserr  GOLONG =BSERR
           00
1451            *
1452 0A09F 70B3  PRGEXT GOSUB  poll            hPURGE SAYS ILL. FILE SPEC
1453 0A0A3 01           CON(2) =pPURGE
1454 0A0A5 43F          GOC    bserr           HANDLED & ERRORED?
1455 0A0A8 831          ?XM=0                  HANDLED OKAY?
1456 0A0AB 60           GOYES  PURG30
1457            * DEFAULT HANDLER - ILLEGAL FILE SPECIFIER
1458 0A0AD 605D         GOTO   FSPCER
1459            *
1460 0A0B1 7D41  PURG30 GOSUB  PRGF40          Purge FIB entry
1461 0A0B5 5AA          GONC   PURGDN          (B.E.T.)
1462            * 'PURGE ALL' COMMAND
1463 0A0B8 847   PURG40 ST=0   7               Clear curr running file flag
1464 0A0BB 715F         GOSUB  C=MAIN
1465 0A0BF 135   PURG45 D1=C
1466 0A0C2 14B          A=DAT1 B
1467 0A0C5 968          ?A=0   B               END OF FILE CHAIN?
1468 0A0C8 39           GOYES  PURG10
1469            * Save start of next file to purge
1470            * Guards against infinite loop
1471            * (Repetitively purging workfile and re-creating it)
1472 0A0CA 1B17         D0=(5) =S-R0-0
           8F2
1473 0A0D1 144          DAT0=C A               UPDATE PTR TO FILE CHAIN
1474 0A0D4 D5           B=C    A
1475            * CHECK PROTECTION
1476 0A0D6 8E00         GOSUBL =GETPR1
           00
1477 0A0DC 832          ?SB=0
1478 0A0DF B0           GOYES  PURG47             NO PURGE PROTECT?
1479            * FILE IS PURGE PROTECTED - SCAN TO NEXT FILE
1480 0A0E1 8E00         GOSUBL =FLSKPB
           00
1481 0A0E7 57D          GONC   PURG45          (B.E.T.)
1482            * FILE NOT PROTECTED - OKAY TO PURGE
1483            * PURGE FILE & ADJUST ALL POINTERS
1484 0A0EA 84B   PURG47 ST=0   11              Purging file in MAIN
1485 0A0ED 7470         GOSUB  PRGF
1486 0A0F1 47A          GOC    bserr           Only errors if not enough mem
1487            * Returns with D0 at RSTKBp (RSTK<R)
1488 0A0F4 1B17         D0=(5) =S-R0-0
           8F2
1489 0A0FB 146          C=DAT0 A               RESTORE PTR TO FILE CHAIN
1490 0A0FE 50C          GONC   PURG45          (B.E.T.)
1491            *
```

```
1492                *
1493 0A101 14A     =eolxc+ A=DAT0 B
1494 0A104 8C00    =eolxck GOLONG =EOLXCK
           00
1495                *
1496                *
1497                *
1498                ********************************************************************
1499                ********************************************************************
1500                **
1501                ** Name:(S) PRGFMF  -  Purge File in Memory
1502                **
1503                ** Category:   FILUTL
1504                **
1505                ** Purpose:
1506                **      Purges specified file
1507                **
1508                ** Entry:  2 entry points:
1509                **          1) PRGFMF - D(S) as it is after FINDF call
1510                **             D1 pointing to start of file header
1511                **          2) PRGF   - File in MAIN; S11=0.
1512                **             D1 at file type in file header.
1513                **
1514                ** Exit:
1515                **      Carry set => error# loaded in C(3-0)
1516                **                   Caller should exit using BSERR
1517                **      Carry clr => File purged successfully
1518                **                   S7=1 => Purged current running file
1519                **
1520                ** Calls:      POLL, RAMROM, GETPRO, EOFLCH, CREATF, FINDWF
1521                **             LEXBF+, ZERPGM, MEMCKL, PUGFIB, FILSKP,
1522                **             RFA-I, D1=CRS, RSTOFS, MOVEUM, EDIT81
1523                **
1524                ** Uses.......
1525                **  Exclusive: A-D, D0,D1,R0,R1,S-R0-0,S-R0-1, S7,S9-S11
1526                **    If purging current file, also uses R2 & R3, S6,S8, S-R0-0
1527                **    If purging a LEX file, also use R2,R3
1528                **    If purging current file AND there's no workfile, uses S0-S7
1529                **
1530                ** Stk lvls:   5
1531                **
1532                **    Date      Programmer   Modifications
1533                **  --------    ----------   -------------
1534                **  08/04/82    S.W.         Added documentation
1535                **  12/16/82    S.W.         Replaced calls to RSTK=R and
1536                **                            R=RSTK with R<RSTK and RSTK<R
1537                **                            C(S) now used
1538                **  06/06/83    S.W.         Replaced call to CLSUSP with
1539                **                            a call to ZERPGM. (Poll must
1540                **                            go out when curr file purged)
1541                **
1542                ********************************************************************
1543                ********************************************************************
1544                *
1545                ********************************************************************
```

```
1546            ****************************************************************
1547            **
1548            ** Name:(S) pPRGPR  -  Poll to PURGE file on non-RAM device
1549            **
1550            ** Category:   POLL
1551            **
1552            ** Type:       POLL
1553            **
1554            ** Purpose:
1555            **      Polls for PURGE of file on non-RAM memory device
1556            **
1557            ** Should poll be "Handled" (return with XM=0)?:
1558            **      Yes
1559            **
1560            ** Meaning of "Handling" Poll (what does code do if handled?):
1561            **      Checks File Protection.
1562            **      If current file, ensure there's a workfile in mainframe
1563            **       or room to create one - See Note below.
1564            **      If not secure, purge the file.
1565            **      Call RFAD-I with begin source in R0, offset in B(A)
1566            **       and D1 pointing to S-R0-1 (which contains old enf of
1567            **       file chain)
1568            **      Have S10 set on exit iff a LEX file was purged.
1569            **      S9 should be set on return iff current file purged
1570            **
1571            **      Mainframe will handle:
1572            **       Deleting any associated FIB.
1573            **       If current file purged (S9=1), SUSP annun. will be
1574            **        cleared & new workfile created.
1575            **       If LEX file purged (S10=1), will call LEXBF+
1576            **       If current running file purged, S7 will be set.
1577            **
1578            ** Entry conditions for handler (registers, ST, RAM, etc.):
1579            **      Carry clear
1580            **      B[A] = Poll number.
1581            **      HEX mode.
1582            **      P=0.
1583            **      D1 at file header of file to purge
1584            **      D(S) contains device type
1585            **       2 => ROM
1586            **       3 => EPROM
1587            **      D(B) contains port info
1588            **
1589            ** Normal exit conditions from handler if handled (ST, RAM,
1590            ** registers, etc.):
1591            **      Carry clear.
1592            **      HEX mode.
1593            **      XM=0.
1594            **      PCADDR intact
1595            **      S9 set iff current file purged
1596            **      S10 set if LEX file purged
1597            **
1598            ** Normal exit conditions from handler if not handled (ST, RAM,
1599            ** registers, etc.):
1600            **      Carry clear
```

```
1601            **      HEX mode.
1602            **      XM=1.
1603            **
1604            ** Error exit conditions from handler
1605            **      Carry set.
1606            **      HEX mode.
1607            **      C[0-3] = Error number.
1608            **      Possible errors are:
1609            **        eFPROT (file is SECURE)
1610            **        eMEM   (file is current, there's no workfile, and no
1611            **                room to create one)
1612            **
1613            ** Available subroutine levels:
1614            **      6
1615            **
1616            ** NOTE:
1617            **      If file to purge is current file, consult mainframe code
1618            **       between PRGF25 & PRGF35 to verify there's a workfile
1619            **       or room to create one.
1620            **
1621            **      --SCRATCH RAM TO CONSIDER BELOW:--
1622            **      --STMT/FN Scratch, SCRTCH, SNAPBF, TRFMBF, LDCSPC,--
1623            **      --LEXPTR.--
1624            **
1625            ** What registers/RAM may be used if handled?:
1626            **      A-D, D0, D1, P
1627            **      R0,R1,R2,R3,S-R0-0,S-R0-1,S0-S7,S9-S11
1628            **
1629            ** What registers/RAM may be used if not handled?:
1630            **      A-D, D0, D1, P
1631            **      Same as if handled (see above)
1632            **
1633            ** What registers/RAM may be used if error exit:
1634            **      A-D, D0, D1, P
1635            **      Same as if handled (see above)
1636            **
1637            ** Special memory/pointer considerations (are pointers funny?):
1638            **      No
1639            **      However, it is important to consider that PURGE of
1640            **       current file CAN and SHOULD generate an insufficient
1641            **       memory error if there's no workfile in the mainframe &
1642            **       no room to create one.
1643            **
1644            ** Envisioned application(s):
1645            **      PURGE A:PORT(2) ! where PORT#2 is an EPROM
1646            **
1647            ** Note:
1648            **      If no one responds, error generated is eFACCS
1649            **
1650            ** History:
1651            **
1652            **    Date     Programmer            Modification
1653            **  --------   ----------   ----------------------------------
1654            **  06/29/82   S.W.         Added documentation
1655            **  12/16/82   S.W.         Eliminated check to distinguish
```

```
1656            **                              ROM from other non-RAM devices
1657            **  12/16/82   S.W.            Poll handler no longer requires
1658            **                              entry point to PRGF40
1659            **  06/03/83   S.W.            Replaced call to CLSUSP with ZERPGM
1660            **
1661            ********************************************************************
1662            ********************************************************************
1663            *
1664            * POLL - PURGE on non-RAM memory device
1665 0A10A 7543  PRGRO+ GOSUB  poll
1666 0A10E 23          CON(2) =pPRGPR
1667 0A110 400         RTNC
1668 0A113 831         ?XM=0                   HANDLED?
1669 0A116 A0          GOYES  PRGRTN
1670 0A118 3300 =PRGFE LC(4)  =eFACCS
          00
1671 0A11E 02          RTNSC
1672            *
1673 0A120 66D0  PRGRTN GOTO   PRGF37          Adjust ptrs,etc;rtn w/carry clr
1674            *
1675 0A124 1CF   PRGIRM D1=D1- 16
1676 0A127 137         CD1EX
1677 0A12A 134         DO=C                    SAVE PTR TO HEADER
1678 0A12D 8E00        GOSUBL =EOFLCH          PT TO END OF CHAIN
          00
1679 0A133 E6          C=C+1  A
1680 0A135 E6          C=C+1  A                PT PAST 00 BYTE
1681 0A137 132         ADOEX
1682 0A13A 131         D1=A                    RESTORE PTR TO HEADER
1683 0A13D 17F         D1=D1+ 1FNAMh
1684 0A140 85B         ST=1   11               File to purge on plug-in
1685 0A143 5B2         GONC   PRGF20           (B.E.T.)
1686            *
1687 0A146 847  =PRGFMF ST=0   7                Curr running file flag
1688 0A149 7AA4        GOSUB  RAMROM
1689 0A14D 5CB         GONC   PRGRO+
1690            *
1691 0A150 8E00        GOSUBL =GETPR1
          00
1692 0A156 832         ?SB=0
1693 0A159 40          GOYES  PRGF05           NO PURGE PROTECT?
1694 0A15B 02          RTNSC
1695 0A15D 868   PRGF05 ?ST=0  8                Not in MAIN?
1696 0A160 4C          GOYES  PRGIRM
1697 0A162 84B         ST=0   11               File in MAIN
1698            * ENTRY PT ASSUMING PROT. ALLOWS PURGE - ASSUMES D1 @ FILE HEADER
1699 0A165 1B49  PRGF   DO=(5) =AVMEMS
          5F2
1700 0A16C 146         C=DATO A
1701 0A16F 1B67  PRGF20 DO=(5) =S-RO-1
          8F2
1702 0A176 144         DATO=C A                UTILITY 'AVMEMS'
1703 0A179 85A         ST=1   10               PURGING LEX FILE FLAG
1704 0A17C D0          A=0    A
1705 0A17E 15B3        A=DAT1 4                READ IN FILE TYPE
```

```
1706 0A182 3480          LC(5)  =fLEX
           2E0
1707 0A189 1CF           D1=D1- 1FNAMh
1708 0A18C 8A2           ?A=C   A              LEX FILE?
1709 0A18F 50            GOYES  PRGF25
1710 0A191 84A           ST=0   10             NOT A LEX FILE
1711               *
1712 0A194 137   PRGF25 CD1EX
1713 0A197 D7            D=C    A              BGN DEST.
1714 0A199 8E00          GOSUBL =FILSKP
           00
1715 0A19F 108           RO=C                  BGN SOURCE
1716 0A1A2 DB            C=D    A              Bgn Dest.
1717 0A1A4 DA            A=C    A              Bgn Dest.
1718 0A1A6 849           ST=0   9              Current File flag
1719 0A1A9 8E00          GOSUBL =D1=CRS        Set C(A) & D1 to CURRST
           00
1720 0A1AF 8A6           ?A#C   A              Not purging current file?
1721 0A1B2 F2            GOYES  PRGF35
1722               * Purging current file - ensure there's enough memory left
1723               *  in MAIN to create new workfile
1724 0A1B4 859           ST=1   9              Purging current file
1725               * See if new workfile will need to be created
1726               *  If not, don't bother with special checks
1727 0A1B7 7EED          GOSUB  FINDWF
1728 0A1BB 552           GONC   PRGF35         Workfile Already Exists?
1729               * Will need to create workfile- ensure there's ample memory
1730 0A1BE 7890          GOSUB  RSTOFS         Restore offset (-file len)
1731 0A1C2 D2            C=0    A
1732 0A1C4 3113          LC(2)  (oFLSTr)       Total amt of memory needed
1733 0A1C8 87B           ?ST=1  11             File to purge on plug-in?
1734 0A1CB C0            GOYES  PRGF27
1735               * Subtract off length of file being purged
1736 0A1CD C9            C=C+B  A              Carries if file being purged
1737 0A1CF 511           GONC   PRGF35          > size of new workfile+LEEWAY
1738 0A1D2 8AA           ?C=0   A
1739 0A1D5 C0            GOYES  PRGF35
1740               * C(A) contains amt of memory required (counting leeway)
1741               * Only 2 cases fall into here:
1742               *  1) Current file on plug-in
1743               *  2) Current file non-BASIC
1744 0A1D7 8F00   PRGF27 GOSBVL =MEMCKL
           000
1745 0A1DE 400           RTNC                  Not enough mem to purge curr
1746               * Restore A(A),B(A); Set D1
1747 0A1E1 7570   PRGF35 GOSUB  RSTOFS         Set A(A),B(A),D1
1748               * B(A)=offset; D1=Begin Dest; A(A)=Begin Source
1749 0A1E5 8F00          GOSBVL =MOVEU1
           000
1750               * Begin source in RO
1751 0A1EC 1F67          D1=(5) =S-RO-1        Pointer to ptr to end source
           8F2
1752 0A1F3 7264          GOSUB  RFAD-I         ADJ FOR-NXT & GOSUB STKS
1753 0A1F7 86A   PRGF37 ?ST=0   10             DIDN'T PURGE LEX FILE?
1754 0A1FA 80            GOYES  PRGF40
```

```
1755                 * ASSUMES LEXBF+ WON'T USE >4 SUBR LEVELS
1756 0A1FC 8E00          GOSUBL =LEXBF+          DELETE LEX TABLE ENTRY
             00
1757 0A202 8E00   PRGF40 GOSUBL =PUGFIB
             00
1758 0A208 869           ?ST=0  9                Current file not purged?
1759 0A20B D4            GOYES  PRGF85
1760                 * Clear any 'run-time memory' which this program might
1761                 *  have sitting around
1762 0A20D 8E00          GOSUBL =ZERPGM
             00
1763                 *
1764                 * Change CURRST & CURREN TO <workfile>
1765                 *
1766 0A213 85A           ST=1   10               PURGE FLAG
1767 0A216 7F8D          GOSUB  FINDWF
1768 0A21A 5D4           GONC   PRGF90           Workfile exists?
1769                 * Create new workfile - we know there's room
1770 0A21D 102           R2=A                    Save name for EDIT79 call
1771 0A220 D2            C=0    A
1772 0A222 3113          LC(2)  =oFLSTr
1773 0A226 108           R0=C                    Save size for CREATF
1774 0A229 D5            B=C    A
1775 0A22B 1F49          D1=(5) =AVMEMS
             5F2
1776 0A232 143           A=DAT1 A                Set up A(A), D1 for CREATF
1777 0A235 8E00          GOSUBL =CRETM5          Open memory,adjust ptrs
             00
1778 0A23B 7F23          GOSUB  EDIT79           Write out header info;move ptrs,etc
1779                 * workfile CREATED - SEE IF POINTED TO BY S-R0-0
1780                 * This is to keep PURGE ALL from going into an infinite loop
1781                 * CURRST IN B(A) - CURREN IN C(A)
1782                 * D0 at CURREN
1783 0A23F 1A17          D0=(4) =S-R0-0
             8F
1784 0A245 142           A=DAT0 A
1785 0A248 8A4           ?A#B   A                workfile NOT POINTED TO BY S-R0-0?
1786 0A24B 50            GOYES  PRGF80
1787 0A24D 144           DAT0=C A                Update S-R0-0 (avoid infinite loop)
1788                 *
1789                 * YMDHMS CLEARS S7
1790 0A250 86D    PRGF80 ?ST=0  13               NOT RUNNING FROM PGM MEM?
1791 0A253 50            GOYES  PRGF85
1792 0A255 857           ST=1   7                PURGED RUNNING PROGRAM FLAG
1793                 *
1794 0A258 03     PRGF85 RTNCC
1795                 *
1796 0A25A DB     RSTOFS C=D    A                Begin Dest
1797 0A25C 135           D1=C
1798 0A25F 110           A=R0                    Begin Source
1799 0A262 D5            B=C    A                Begin Dest
1800 0A264 E8            B=B-A  A                Begin Dest-Begin Source
1801 0A266 01            RTN                     B(A) contains offset
1802                 *
1803 0A268 7943   PRGF90 GOSUB  EDIT81           Pos ptrs at workfile
```

```
1804 0A26C 53E          GONC   PRGF80          (B.E.T.)
1805              *
1806              *
```

```
1807                STITLE Continuation of RENAME Execute
1808          ********************************************************************
1809          ********************************************************************
1810          **
1811          ** Name:    RNM010  -  Continuation of RENAME Execution Code
1812          **
1813          ** Category: STEXEC
1814          **
1815          ** Purpose:  Continuation of RENAME Execution
1816          **
1817          ** Entry:    DO past the RENAME tokenization
1818          **           P=0
1819          **           Filespec source and destination information
1820          **            has been saved on the SAVSTK. (See RENAME
1821          **            execute for details).
1822          **
1823          ** Exit:     via NXTSTM
1824          **
1825          ** Calls:    FINDF, tKYSck, FSPECx, POLL, ALINFO, DEFFIL,
1826          **           RDINFO, RAMROM, LAKEYS, BASKEY, SVINFO
1827          **
1828          ** Uses:
1829          **   exclusive...  SAVSTK, A-D, D1,D0
1830          **   inclusive...  A-D, D1,D0, R0-R3, S1,S2,S7
1831          **                 STMTD0, STMTR1 (All of it)
1832          **                 All of function scratch
1833          **
1834          ** Detail:     RENAME file TO KEYS renames the specified file &
1835          **             makes it the active key assignment file.  It
1836          **             follows that if the specified file to rename is
1837          **             not a key file, an error will occur.
1838          **
1839          **             Mainframe-only keywords such as KEYS are
1840          **             considered to specify a source, therefore any
1841          **             device given on the destination is ignored.
1842          **
1843          ** Note:       The actual start of RENAME execution code is in
1844          **             the JP&EXC module; it shares some common code with
1845          **             COPY, then jumps off to the RNM010 entry point in
1846          **             this module.
1847          **
1848          ** Stack lvls: 7
1849          **
1850          ** History:
1851          **
1852          **    Date     Programmer   Modification
1853          **  --------   ----------   ------------
1854          **  06/29/82   S.W.         Added documentation.
1855          **  07/29/82   S.W.         Added special check to
1856          **                           RENAME <file> TO KEYS to ensure
1857          **                           specified file is in the mainframe.
1858          **
1859          ********************************************************************
1860          ********************************************************************
1861          *
```

```
1862            ****************************************************************
1863            ****************************************************************
1864            **
1865            ** Name:(S) pRNAME  -  Poll to RENAME file on unknown device
1866            **
1867            ** Category:   POLL
1868            **
1869            ** Type:       POLL
1870            **
1871            ** Purpose:
1872            **      Polls to RENAME file on external device or on non-RAM
1873            **      memory device.
1874            **
1875            ** Should poll be "Handled" (return with XM=0)?:
1876            **      Yes
1877            **
1878            ** Meaning of "Handling" Poll (what does code do if handled?):
1879            **      Writes out new name to file header (or directory)
1880            **       Should be ready to go on to NXTSTM
1881            **
1882            ** Entry conditions for handler (registers, ST, RAM, etc.):
1883            **      Carry clear
1884            **      B[A] = Poll number.
1885            **      HEX mode.
1886            **      P=0.
1887            **      D0 is past the file specifier
1888            **      Proposed new file name is in the SAVSTK area
1889            **       (or at least what WAS the SAVSTK area before poll)
1890            **       The 10 character blank-filled new file name is 112
1891            **       nibbles LOWER in memory than where SAVSTK points
1892            **       (70 HEX).
1893            **
1894            **      D(S) >= 7 =>
1895            **              RENAME file on external device
1896            **              Name of file to rename is blank-filled in A(W);
1897            **               Characters 9 & 10 in R0
1898            **              D(S),D(X) contain device id
1899            **              In higher memory, adjacent to proposed file name
1900            **               given above, is its corresponding 5 nibble
1901            **               device id (Do a shift right circular to restore
1902            **               to original form).
1903            **              If poll isn't handled, default error is eFSPEC
1904            **
1905            **      D(S) < 7  =>
1906            **              RENAME file on non-RAM memory device
1907            **              D1 is at the file header
1908            **              D(S) contains memory type info
1909            **                  1 => ROM
1910            **                  2 => EPROM
1911            **              D(B) contains port number/extender
1912            **              If poll isn't handled, default error is eFACCS
1913            **
1914            ** Normal exit conditions from handler if handled (ST, RAM,
1915            ** registers, etc.):
1916            **      Carry clear
```

```
1917            **      HEX mode.
1918            **      XM=0.
1919            **      Ready to go on to NXTSTM
1920            **
1921            ** Normal exit conditions from handler if not handled (ST, RAM,
1922            ** registers, etc.):
1923            **      Carry clear
1924            **      HEX mode.
1925            **      XM=1.
1926            **      R0 intact from entry.
1927            **
1928            ** Error exit conditions from handler:
1929            **      Carry set.
1930            **      HEX mode.
1931            **      C[0-3] = Error number.
1932            **
1933            ** Available subroutine levels:
1934            **      7
1935            **
1936            ** NOTE:
1937            **      Error if:
1938            **       1) No file name is specified, ie RENAME A TO :TAPE
1939            **            (eFSPEC)
1940            **          (This is default error given if D(S)>=7, else
1941            **            the handler MUST EXPLICITLY error)
1942            **       2) Proposed file name is 'keys'
1943            **            (eFSPEC)
1944            **       3) File by that name already exists on the medium
1945            **            (eFEXST)
1946            **
1947            ** What registers/RAM may be used if handled?:
1948            **      A-D, D0, D1, P
1949            **      R0-R4, S0-S11, STMT/FN scratch
1950            **
1951            ** What registers/RAM may be used if not handled?:
1952            **      A-D, D0, D1, P
1953            **      R1-R3, S0-S11, STMT/FN scratch
1954            **      Don't alter SAVSTK !
1955            **
1956            ** What registers/RAM may be used if error exit:
1957            **      A-D, D0, D1, P
1958            **      R0-R4, S0-S11, STMT/FN scratch
1959            **
1960            ** Special memory/pointer considerations (are pointers funny?):
1961            **      No
1962            **
1963            ** Envisioned application(s):
1964            **      RENAME A:<external device> TO B
1965            **      RENAME A:PORT(3) TO B           (where A is on EPROM)
1966            **
1967            ** History:
1968            **
1969            **    Date     Programmer          Modification
1970            **  --------   ----------   ----------------------------------
1971            **  12/16/82   S.W.         Combined polls
```

```
1972            **  05/17/83   S.W.         Added documentation
1973            **
1974            ************************************************************
1975            ************************************************************
1976            *
1977            *
1978 0A26F 06  =RNM010 RSTK=C                  Save Destination info
1979 0A271 843         ST=0    sDEST
1980 0A274 8E00        GOSUBL  =RDINFO
          00
1981 0A27A DB          C=D     A
1982 0A27C B06         C=C+1   P
1983 0A27F 07          C=RSTK                  Restore Destination info
1984 0A281 540         GONC    RNM015          Device specifier on source?
1985            * No device specifier on source
1986 0A284 D7          D=C     A               Take destination device specifier
1987 0A286 817   RNM015 DSRC
1988 0A289 76DC        GOSUB   FINDF+
1989 0A28D 4F5         GOC     RNM085
1990            * File found
1991 0A290 7363        GOSUB   RAMROM
1992 0A294 5D5         GONC    RNMPOL          not in RAM?
1993 0A297 AFB         C=D     W               Copy file chain info
1994 0A29A 10A         R2=C                    Save chain info
1995 0A29D 137         CD1EX
1996 0A2A0 06          RSTK=C                  Save ptr to file to rename
1997 0A2A2 853         ST=1    sDEST
1998 0A2A5 8E00        GOSUBL  =RDINFO         Get proposed name
          00
1999 0A2AB 978         ?A=0    W
2000 0A2AE 86          GOYES   RNM070
2001 0A2B0 11A         C=R2
2002 0A2B3 AF7         D=C     W               Restore file chain info
2003 0A2B6 7DBC        GOSUB   FINDF
2004 0A2BA 5F5         GONC    RNMERR
2005            * Proposed file name in A & B
2006 0A2BD 07          C=RSTK
2007 0A2BF 135         D1=C                    Restore ptr to file to rename
2008 0A2C2 76A0        GOSUB   LAKEYS
2009 0A2C6 974         ?A#B    W               Not renaming to keys?
2010 0A2C9 91          GOYES   RNM050
2011            * Ensure file to rename is in the MAINFRAME
2012            * And that file is a keys file
2013 0A2CB 11A         C=R2
2014 0A2CE A4E         C=C-1   S
2015 0A2D1 544         GONC    RNM070
2016 0A2D4 8E00        GOSUBL  =BASKEY
          00
2017 0A2DA 594         GONC    RNM095          Not a KEY file?
2018 0A2DD 866         ?ST=0   6
2019 0A2E0 44          GOYES   RNM095          Not a KEY file?
2020 0A2E2 AF4   RNM050 A=B     W
2021 0A2E5 159F  RNM055 DAT1=A  1FNAMh
2022 0A2E9 667D  RNM060 GOTO    PURGDN
2023            *
```

```
2024 0A2ED 876   RNM085 ?ST=1  6              File not found
2025 0A2F0 E2           GOYES  mferr
2026             *
2027 0A2F2 7D51  RNMPOL GOSUB  poll
2028 0A2F6 11           CON(2) =pRNAME
2029             * Requirements of this handler:
2030             * Handled => go on to NXTSTM
2031             * D(S)<7  => default is eFACCS
2032             * D(S)>=7 => default is eFSPEC
2033 0A2F8 491   =pPOLL2 GOC    Bserr          Handled and errored?
2034 0A2FB 831          ?XM=0                  Handled Okay?
2035 0A2FE BE           GOYES  RNM060
2036             * Not handled - give correct error
2037 0A300 B47          D=D+1  S
2038 0A303 A47          D=D+D  S               External devices will carry
2039 0A306 4F0          GOC    RNM070
2040 0A309 978          ?A=0   W               No file name specified?
2041 0A30C A0           GOYES  RNM070
2042 0A30E 760E =PRGROM GOSUB  PRGFE           Load eFACCS
2043 0A312 668D  Bserr  GOTO   bserr
2044             *
2045 0A316 67EA  RNM070 GOTO   FSPCER          Illegal Filespec
2046             *
2047 0A31A 3100 =RNMERR LC(2)  =eFEXST         DUPLICATE FILE
2048 0A31E 8C00  mferr  GOLONG =MFERR
           00
2049             *
2050 0A324 65A2  RNM095 GOTO   EDITXE          Illegal file type
2051             *
```

```
2052                    STITLE NAME Execute
2053                ****************************************************************
2054                ****************************************************************
2055                **
2056                ** Name:    NAME    -  Execute NAME Statement
2057                **
2058                ** Category:   STEXEC
2059                **
2060                ** Purpose:
2061                **      Executes NAME Statement
2062                **
2063                ** Entry:
2064                **      P       =  0
2065                **      DO past tNAME
2066                **
2067                ** Exit:
2068                **      P       =  0
2069                **      workfile NAMED
2070                **
2071                **      ERROR exits if
2072                **                      1) File by name given already exists in
2073                **                         the mainframe - eFEXST
2074                **                      2) No workfile - eFnFND
2075                **                      3) Illegal file spec - eFSPEC
2076                **
2077                ** Calls:    FILEF, FILEXQ, WRKFIL
2078                **
2079                ** Uses:
2080                **   Exclusive...   A, B, C, D, D1, DO, S8
2081                **   Inclusive...   A-D, R0-R3, D1,DO, S1,S2,S7, STMTDO,
2082                **                  STMTR1 (All of it), All of function scratch
2083                **
2084                ** Detail:
2085                **           Only allows simple file names (no devices)
2086                **
2087                ** Stk lvls:   6
2088                **
2089                ** History:
2090                **
2091                **    Date      Programmer              Modification
2092                **  --------   ----------   ------------------------------------
2093                **  06/29/82   S.W.         Added documentation
2094                **  12/13/82   S.W.         No longer creates new workfile
2095                **
2096                ****************************************************************
2097                ****************************************************************
2098                *
2099                *
2100 0A328 0000          REL(5) =NAMEDC
           0
2101 0A32D 0000          REL(5) =NAMEP
           0
2102                *
2103                * IF FILE BY THAT NAME DOESN'T ALREADY EXIST, RENAME workfile
2104                *
```

```
2105 0A332 8EE3 =NAME   GOSUBL FILXQ^
           8F
2106 0A338 5DD          GONC   RNMO70          Illegal file name?
2107 0A33B 978          ?A=0   W               No file name specified?
2108 0A33E 8D           GOYES  RNMO70
2109 0A340 B47          D=D+1  S
2110 0A343 52D          GONC   RNMO70          Device specified?
2111             * SEARCH MAIN ONLY FOR POSSIBLE DUPLICATE FILE NAME
2112 0A346 766C         GOSUB  FILEF
2113 0A34A 5FC          GONC   RNMERR          DUPLICATE FILE NAME?
2114             *
2115 0A34D 100          R0=A                   SAVE PROPOSED FILE NAME
2116 0A350 755C         GOSUB  FINDWF
2117 0A354 49C          GOC    mferr           workfile doesn't exist?
2118 0A357 110          A=R0
2119 0A35A 5A8          GONC   RNMO55          (B.E.T.) NAME workfile
2120             *
2121             *
2122             *
2123             ********************************************************************
2124             ********************************************************************
2125             **
2126             ** Name:    tKYSck  -  Check for tKEYS; Loads 'keys    '
2127             **
2128             ** Category:   FILUTL
2129             **
2130             ** Purpose:  tKYSck entry checks for tKEYS token.
2131             **
2132             **           LAKEYS entry simply loads A & C registers with
2133             **           \    syek\.
2134             **
2135             ** Entry:
2136             **           2 entry points:
2137             **           P=0
2138             **           1) tKYSck - DO at possible tKEYS token
2139             **           2) LAKEYS - No additional requirements.
2140             **
2141             ** Exit:
2142             **           LAKEYS entry:
2143             **            Carry preserved from entry
2144             **            D(S)=0
2145             **            C and A contain necessary ascii to search for
2146             **             <keys> file.
2147             **           tKYSck entry:
2148             **            Carry clr => Found tKEYS
2149             **                         DO past tKEYS
2150             **                         C and A contain necessary ascii
2151             **                          to search for <keys> file.
2152             **                         D(S)=0
2153             **            Carry set => DO unchanged
2154             **                         A(B) = token pointed to by DO
2155             **                         C(B) = tKEYS
2156             **
2157             ** Calls:    none
2158             **
```

```
2159            ** Uses:     A, C, D(S)
2160            **
2161            ** Stack lvls: 0
2162            **
2163            ** History:
2164            **
2165            **    Date     Programmer   Modification
2166            **  --------   ----------   ------------
2167            **  06/30/82   S.W.         Added documentation
2168            **
2169            ********************************************************************
2170            ********************************************************************
2171            *
2172            *
2173 0A35D 14A  =tKYSck A=DAT0 B
2174 0A360 3100         LC(2)  =tKEYS
2175 0A364 966          ?A#C   B
2176 0A367 00           RTNYES
2177            *
2178 0A369 161          D0=D0+ 2              STEP OVER TOKEN
2179 0A36C 3FB6 =LAKEYS LCASC  \    syek\
           5697
           3702
           0202
           02
2180 0A37E AC3          D=0    S
2181 0A381 AFA          A=C    W
2182 0A384 01           RTN
```

```
2183                    STITLE SECURE/UNSECURE Execute
2184            ******************************************************************
2185            ******************************************************************
2186            **
2187            ** Name:    SECURE  -  Executes SECURE Statement
2188            ** Name:    UNSECR  -  Executes UNSECURE Statement
2189            **
2190            ** Category: STEXEC
2191            **
2192            ** Purpose:  Executes SECURE/UNSECURE Statements
2193            **
2194            ** Entry:
2195            **
2196            **           DO after tSECURE or tUNSECURE
2197            **           P=0
2198            **           2 ENTRY POINTS:
2199            **            1) SECURE
2200            **            2) UNSECURE
2201            **
2202            ** Exit:
2203            **           P=0
2204            **
2205            ** Uses:
2206            **  Exclusive:  S6,S8,S10, A-D, D1,D0, R1-R3
2207            **  Inclusive:  A-D, D1,D0, R0-R3, S1,S2,S7, STMTD0,
2208            **              STMTR1 (All of it), All of function scratch
2209            **
2210            ** Calls:    GETPRO, FINDF, tKYSck, FSPECx, POLL, EOLXCK,
2211            **           RAMROM, CURDVC, FTYPFD, RDENTY
2212            **
2213            ** Stack lvls: 7
2214            **
2215            ** Detail:     Flag nibble in file header - Bit 0 SECURE
2216            **                                          Bit 1 PRIVATE
2217            **
2218            ** History:
2219            **
2220            **    Date     Programmer   Modification
2221            **  --------   ----------   ------------
2222            **  06/30/82   S.W.         Added documentation
2223            **  12/17/82   S.W.         Errors on ROM
2224            **
2225            ******************************************************************
2226            ******************************************************************
2227            *
2228            ******************************************************************
2229            ******************************************************************
2230            **
2231            ** Name:(S) pFPROT  -  [UN]SECURE or PRIVATE in non-RAM device
2232            **
2233            ** Category:   POLL
2234            **
2235            ** Type:       POLL
2236            **
2237            ** Purpose:
```

```
2238            **      Poll to SECURE/UNSECURE/PRIVATE file on external device
2239            **       or in non-RAM memory device
2240            **
2241            ** Should poll be "Handled" (return with XM=0)?:
2242            **      Yes
2243            **
2244            ** Meaning of "Handling" Poll (what does code do if handled?):
2245            **      Change file protection; ready to go on to NXTSTM
2246            **
2247            ** Entry conditions for handler (registers, ST, RAM, etc.):
2248            **      Carry clear.
2249            **      B[A] = Poll number.
2250            **      HEX mode.
2251            **      P=0.
2252            **      DO past file specification
2253            **      D(S) >= 7 =>
2254            **                File on external device
2255            **                File name blank-filled in A(W);
2256            **                 characters 9 & 10 in low nibbles of R0
2257            **                D(S),D(X) contains device identifier
2258            **                If poll not handled, default error is eFSPEC
2259            **
2260            **      D(S) <  7 =>
2261            **                File in non-RAM memory device
2262            **                D1 at file header
2263            **                D(S) contains memory type info
2264            **                D(B) contains port extender/number
2265            **                If poll not handled, default error is eFACCS
2266            **
2267            **      S11=1 => PRIVATE
2268            **             else
2269            **                S10=1 => UNSECURE
2270            **                    0 => SECURE
2271            **
2272            ** Normal exit conditions from handler if handled (ST, RAM,
2273            ** registers, etc.):
2274            **      Carry clear.
2275            **      HEX mode.
2276            **      XM=0.
2277            **      PCADDR intact (ready to go on to NXTSTM)
2278            **
2279            ** Normal exit conditions from handler if not handled (ST, RAM,
2280            ** registers, etc.):
2281            **      Carry clear.
2282            **      HEX mode.
2283            **      XM=1.
2284            **      S10,S11 intact from entry
2285            **      If D(S)>=7, R0 must be intact from entry
2286            **
2287            ** Error exit conditions from handler:
2288            **      Carry set.
2289            **      HEX mode.
2290            **      C[0-3] = Error number.
2291            **      Only foreseen errors are for PRIVATE on a SECURE or non-
2292            **       executable file, which generates eFPROT, eFTYPE
```

```
2293            **       respectively.
2294            **
2295            ** Available subroutine levels:
2296            **      7
2297            **
2298            ** NOTE:
2299            **      For no file name specified, ie SECURE :<device>
2300            **      if D(S)>=7, the default error for 'not handled' will
2301            **      be eFSPEC.  But if D(S)<7, the handler MUST EXPLICITLY
2302            **      error on this.
2303            **
2304            ** What registers/RAM may be used if handled?:
2305            **      A-D, D0, D1, P,R0-R4
2306            **      STMT/FN Scratch, S0-S11
2307            **
2308            ** What registers/RAM may be used if not handled?:
2309            **      A-D, D0, D1, P
2310            **      R1-R3, S0-S9, STMT/FN Scratch
2311            **      R0 if D(S)<7
2312            **
2313            ** What registers/RAM may be used if error exit :
2314            **      A-D, D0, D1, P
2315            **      R0-R4, S0-S11, STMT/FN scratch
2316            **
2317            ** Envisioned application(s):
2318            **      SECURE A:TAPE
2319            **      PRIVATE A:PORT(3)        where PORT#3 is EPROM
2320            **
2321            ** History:
2322            **
2323            **    Date     Programmer              Modification
2324            **  --------   ----------   ----------------------------------
2325            **  06/30/82   S.W.         Added documentation
2326            **  12/16/82   S.W.         Combined polls
2327            **
2328            ****************************************************************
2329            ****************************************************************
2330            *
2331            *
2332 0A386 0000          REL(5) =SECRDC
         0
2333 0A38B 0000          REL(5) =SECURP
         0
2334 0A390 85A  =UNSECR ST=1   10
2335 0A393 6D00          GOTO   SECURE
2336            *
2337 0A397 0000          REL(5) =SECRDC
         0
2338 0A39C 0000          REL(5) =SECURP
         0
2339            *
2340 0A3A1 7C5D =SECURE GOSUB  eolxc+
2341 0A3A5 590          GONC   SECR02          EOL NOT FOUND?
2342            * NO FILE NAME GIVEN - APPLIES TO CURRENT FILE
2343 0A3A8 7F52          GOSUB  CURDVC          GET DEVICE TYPE OF CURR FILE
```

```
2344 0A3AC 571          GONC   SECR15         (B.E.T.)
2345              *
2346 0A3AF 7AAF  SECR02 GOSUB  tKYSck
2347 0A3B3 590          GONC   SECR10         tKEYS?
2348              *
2349 0A3B6 737B         GOSUB  FSPECx
2350 0A3BA 402          GOC    SECERR
2351 0A3BD 72AB  SECR10 GOSUB  FINDF+
2352 0A3C1 456          GOC    SECEXT         FILE NOT FOUND?
2353 0A3C4 7F22  SECR15 GOSUB  RAMROM
2354 0A3C8 536          GONC   SECPOL         NOT IN RAM?
2355              *
2356 0A3CB 86B          ?ST=0  11             Not PRIVATE command?
2357 0A3CE 92           GOYES  SECR60
2358              * Do not allow PRIVATE on SECURE file
2359 0A3D0 8E00         GOSUBL =GETPR1
           00
2360 0A3D6 832          ?SB=0                 not SECURE?
2361 0A3D9 60           GOYES  SECR20
2362 0A3DB 6DBC  SECERR GOTO   bserr
2363              *
2364 0A3DF 8E00  SECR20 GOSUBL =FTYPFD
           00
2365 0A3E5 5D0          GONC   PRVT20         type unknown?
2366 0A3E8 8E00         GOSUBL =RDENTY
           00
2367 0A3EE 90E          ?C#0   P
2368 0A3F1 90           GOYES  SECR65         executable?
2369 0A3F3 66D1  PRVT20 GOTO   EDITXE         Illegal file type
2370              *
2371 0A3F7 17F   SECR60 D1=D1+ 1FNAMh         Point to file type field
2372 0A3FA 173   SECR65 D1=D1+ 1FTYPh         POINT TO FLAG FIELD
2373 0A3FD 14B          A=DAT1 B              Read in current protection
2374 0A400 302          LCHEX  2              (2 nib read costs less ROM)
2375 0A403 87B          ?ST=1  11             PRIVATE?
2376 0A406 A0           GOYES  SECR70
2377              * SECURE or UNSECURE
2378 0A408 87A          ?ST=1  10             UNSECURE FILE?
2379 0A40B D0           GOYES  UNSEC
2380              * SECURE FILE
2381 0A40D 301          LCHEX  1
2382 0A410 0E0E  SECR70 A=A!C  P              SET LOW BIT
2383 0A414 6A00         GOTO   SECR80
2384              *
2385 0A418 30E   UNSEC  LCHEX  E
2386 0A41B 0E06         A=A&C  P              CLR LOW BIT
2387 0A41F 1590  SECR80 DAT1=A 1              WRITE OUT NEW PROTECTION
2388 0A423 6C3C         GOTO   PURGDN         CONTINUE EXECUTION
2389              *
2390 0A427 876   SECEXT ?ST=1  6
2391 0A42A 1B           GOYES  SECERR         File not found?
2392              *
2393              * POLL FOR SECURE ON PROM
2394 0A42C 7320  SECPOL GOSUB  poll
2395 0A430 B0           CON(2) =pFPROT        Default handlers are eFSPEC &
```

```
2396 0A432 65CE          GOTO   pPOLL2          eFACCS
2397               *
```

```
2398                    STITLE PRIVATE Statement Execute
2399                ********************************************************************
2400                ********************************************************************
2401                **
2402                ** Name:    PRIVAT  -  Executes PRIVATE statement
2403                **
2404                ** Category: STEXEC
2405                **
2406                ** Purpose:  Executes PRIVATE Statement
2407                **
2408                ** Entry:    DO past tPRIVATE
2409                **
2410                ** Exit:     through NXTSTM
2411                **
2412                ** Calls:    FINDF, FSPECx, FTYPFD, POLL, RAMROM, GETPRO
2413                **           RDENTY, tKYSck
2414                **
2415                ** Uses:
2416                **  Exclusive:  A-D, S6,S8,S11, D1,D0, R0-R3
2417                **  Inclusive:  A-D, D1,D0, R0-R3, S1,S2,S7, STMTD0,
2418                **              STMTR1 (All of it), All of function scratch
2419                **
2420                ** Stack lvls: 7
2421                **
2422                ** Detail:   SINCE PRIVACY ONLY MAKES SENSE ON AN EXECUTABLE
2423                **           FILE, THE COMMAND ERRS IF 1) THE FILE TYPE IS
2424                **           UNKNOWN OR 2) IF THE FILE TYPE IS NON-EXECUTABLE.
2425                **
2426                ** History:
2427                **
2428                **    Date     Programmer   Modification
2429                **  --------   ----------   ------------
2430                **  06/30/82   S.W.         Added documentation
2431                **  01/10/83   S.W.         Combined PRIVATE with SECURE;
2432                **                           Combined polls
2433                **  02/08/83   S.W.         No longer can a SECURE file be
2434                **                           made PRIVATE.
2435                **
2436                ********************************************************************
2437                ********************************************************************
2438                *
2439 0A436 0000         REL(5) =NAMEDC
           0
2440 0A43B 0000         REL(5) =PRIVTP
           0
2441 0A440 85B  =PRIVAT ST=1   11
2442 0A443 6B6F         GOTO   SECR02
2443                *
2444 0A447 8C00 =romchk GOLONG =ROMCHK
           00
2445                *
2446 0A44D 8C00 =romfnd GOLONG =ROMFND
           00
2447                *
2448                *
```

```
2449                    STITLE SHOW PORT
2450                ********************************************************************
2451                ********************************************************************
2452                **
2453                ** Name:    SHOW    -  Executes SHOW (PORT) statement
2454                **
2455                ** Category:   STEXEC
2456                **
2457                ** Purpose:
2458                **      Displays information on all ports - type of mem
2459                **      and size
2460                **
2461                ** Entry:
2462                **      P       =  0
2463                **      D0 at numeric expression
2464                **
2465                ** Exit:
2466                **      P       =  0
2467                **
2468                ** Calls:       ROMFND, AVM2DS, CAT$90, MSIZE, CAT$83, S-CRLF,
2469                **              BLNKC+, ROMCHK, PRT#DC, OUTC15, OUTNBS
2470                **
2471                ** Uses.......
2472                **              A-D, D0, D1, R0,R1
2473                **
2474                ** Stk lvls:    MAX(4,1 more than 'several' [S-CRLF])
2475                **
2476                ** Note:        There is an external subroutine to search a
2477                **              device table for the corresponding ascii text.
2478                **              PT-TYF - Port Type Find
2479                **
2480                ** History:
2481                **
2482                **    Date      Programmer              Modification
2483                **  --------    ----------    ----------------------------------
2484                **  08/09/82    S.W.          Wrote routine
2485                **  04/25/83    S.W.          No longer: a) programmable,
2486                **                             b) outputs header, c) outputs device
2487                **                             name (outputs device# instead), or
2488                **                             d) allows for selective PORT.
2489                **  06/10/83    S.W.          Replaced calls to LDCSET & BLANKC
2490                **                             with call to BLNKC+.
2491                **
2492                ********************************************************************
2493                ********************************************************************
2494                *
2495                *
2496                *
2497 0A453         =POLLj
2498 0A453 8C00    poll    GOLONG =POLL
           00
2499                *
2500 0A459 3100    =DVCNFE LC(2)   =eDVCNF
2501 0A45D 60CE            GOTO    nferr
2502                *
```

```
2503                *
2504 0A461 0000     
           0                REL(5) =SHOWp
2505                *
2506 0A466 161      =SHOW    D0=D0+ 2              STEP OVER tPORT
2507 0A469 7ADF              GOSUB  romchk
2508 0A46D 4BE               GOC    DVCNFE         No Ports ?
2509                *D(S)=DEVICE TYPE;D(7-0) CONTAINS SIZE INFO
2510 0A470 AFB       SHOW10 C=D    W
2511 0A473 108               R0=C
2512 0A476 8E00              GOSUBL =BLNKC+        Call LDCSET,BLANKC
           00
2513 0A47C AFA               A=C    W               (B(A)=D0)
2514 0A47F 8E00              GOSUBL =OUTC15
           00
2515 0A485 29                P=     9
2516 0A487 8E00              GOSUBL =OUTNBS        BLANK FILL 26 NIBS
           00
2517 0A48D D9                C=B    A
2518 0A48F 134               D0=C                 Back up D0
2519 0A492 118               C=R0
2520 0A495 D7                D=C    A
2521 0A497 8E00              GOSUBL =PRT#DC
           00
2522 0A49D 118               C=R0
2523 0A4A0 AD0               A=0    M
2524 0A4A3 8E00              GOSUBL =MSIZE+
           00
2525 0A4A9 F0                ASL    A
2526 0A4AB F0                ASL    A
2527 0A4AD 8E00              GOSUBL =CAT$83
           00
2528 0A4B3 161               D0=D0+ 2              LEAVE BLANK BETWEEN FIELDS
2529                * Convert Device# in R0 to decimal & output, suppressing leading
2530                *  zeroes
2531 0A4B6 110               A=R0
2532 0A4B9 D0                A=0    A
2533 0A4BB 810               ASLC
2534 0A4BE 2C                P=     12             Convert to decimal & output up to
2535 0A4C0 8E00              GOSUBL =CAT$90         2 digits
           00
2536 0A4C6 31FF              LCHEX  FF
2537 0A4CA 14C               DATO=C B
2538 0A4CD 8F00              GOSBVL =AVM2DS
           000
2539 0A4D4 8F00              GOSBVL =S-CRLF        SEND BUFFER TO DISPLAY & BUILD
           000
2540 0A4DB 7E6F              GOSUB  romfnd
2541 0A4DF 509               GONC   SHOW10         More ports ?
2542 0A4E2 6D7B              GOTO   PURGDN
2543                *
2544                *
```

```
2545                    STITLE EDIT Execute
2546            ********************************************************************
2547            ********************************************************************
2548            **
2549            ** Name:    EDIT    -  Moves EDIT Pointers to Specified File
2550            ** Name:(S) EDITWF  -  Designates workfile as Current File
2551            ** Name:(S) EDIT80  -  Designates Specified File as Current
2552            ** Name:    EDIT20  -  Collapses Stks; Spec. File Becomes Curr.
2553            **
2554            ** Category: FILUTL
2555            **
2556            ** Purpose:
2557            **           EDIT executes the EDIT statement.
2558            **
2559            **           EDITWF designates the workfile as the current file.
2560            **           If it doesn't exist, it is created.  EDITWF is
2561            **           called when current file is purged and during
2562            **           configuration.
2563            **
2564            **           EDIT80 designates the specified file as current.
2565            **           If file isn't BASIC, a POLL goes out, resulting
2566            **           in an error if no one responds.  This entry
2567            **           point is used by CAT when [f][EDIT] is hit
2568            **           during a multiple file catalog.
2569            **
2570            **           EDIT20 collapses all the execution stacks before
2571            **           designating the specified file as current.  This
2572            **           is the entry point used by RUN.  An assumption is
2573            **           made that this file is of legal type to be made
2574            **           current.
2575            **
2576            ** Entry:    4 entry points:
2577            **           P=0
2578            **             1) EDIT   - D0 past tEDIT.
2579            **
2580            **             2) EDITWF - S10=1 => No collapse of stacks and
2581            **                                  no CATalog.
2582            **                         S10=0 => No collapse of stacks
2583            **                                  CATalog iff S11=0
2584            **
2585            **             3) EDIT80 - S10 and S11 as with EDITWF.
2586            **
2587            **             4) EDIT20 - D1 points at new current file.
2588            **
2589            ** Exit:
2590            **           CURRL UPDATED; Stacks, etc collapsed via CLPSTK
2591            **             Error Exits if:
2592            **               1) file must be created and not enough memory
2593            **               2) specified file is not BASIC
2594            **               3) port# specified that doesn't exist
2595            **               4) non-mainframe device specified
2596            **           If no CATalog is done:
2597            **             B(A)=CURRST; C(A)=D(A)=CURREN;
2598            **             D0 points to CURREN RAM location
2599            **
```

```
2600            ** Calls:    CRETF, CLPSTK, FINDF, SAVEL, WRKFIL
2601            **           EOLXCK, FSPECx, POLL, NULLP, BASKEY
2602            **
2603            ** Uses:     A-D, R0-R3, S6, S8, S9, S10, S11, D1,D0
2604            **           + If FSPECx is called: S1,S2,S7, STMTD0,
2605            **                STMTR1 (All of it), All of function scratch
2606            **
2607            ** Detail:   EDIT is a system command (non-programmable).  The
2608            **           reason for this limitation is that EDIT changes
2609            **           CURRST & CURREN; this would be nonsensical during a
2610            **           running program, since the same pointers are used
2611            **           to indicate current EDIT file as current RUN file.
2612            **
2613            **           EDIT [filename]
2614            **
2615            ** Stack lvls: 7
2616            **
2617            ** History:
2618            **
2619            **    Date     Programmer   Modifications
2620            **  --------   ----------   ------------
2621            **  06/30/82   S.W.         Added documentation
2622            **  07/20/82   S.W.         No longer saves 2 stack levels
2623            **                           (burden put on PRGFMF)
2624            **  09/17/82   J.P.         Set S9 before NULLP call
2625            **  11/11/82   S.W.         Deleted poll on external file
2626            **  12/17/82   S.W.         Eliminated call to CHAIN - caused
2627            **                           problems when old EDIT file is
2628            **                           in non-RAM medium; ptr to new
2629            **                           CURRST no longer in R3 on exit.
2630            **
2631            **  01/11/83   J.P.         Change S9=1 to P=1 before NULLP
2632            **                           call.
2633            **  03/02/83   J.P.         Added pEDIT poll
2634            **
2635            ********************************************************************
2636            ********************************************************************
2637            *
2638            ****************************************************************
2639            ****************************************************************
2640            **
2641            ** Name:(S) pEDIT   -  Poll to position at non-BASIC file
2642            **
2643            ** Category:   POLL
2644            **
2645            ** Type:       POLL
2646            **
2647            ** Purpose:
2648            **      Just gives the 'OK' to position at non-BASIC file
2649            **
2650            ** Should poll be "Handled" (return with XM=0)?:
2651            **      Yes
2652            **
2653            ** Meaning of "Handling" Poll (what does code do if handled?):
2654            **      Clears XM
```

```
2655            **
2656            ** Entry conditions for handler (registers, ST, RAM, etc.):
2657            **      B[A] = Poll number.
2658            **      HEX mode.
2659            **      P=0.
2660            **      D1 points at file header
2661            **      A(A) contains file type#
2662            **
2663            ** Normal exit conditions from handler if handled
2664            **      Carry clear (POLL only).
2665            **      HEX mode.
2666            **      XM=0.
2667            **      D1 at file header
2668            **      S11 preserved from entry (flags whether to CATalog)
2669            **      P=0
2670            **
2671            ** Normal exit conditions from handler if not handled (ST, RAM,
2672            ** registers, etc.):
2673            **      Carry clear (POLL only).
2674            **      HEX mode.
2675            **      XM=1.
2676            **      S11 preserved from entry
2677            **      P=0
2678            **
2679            ** Error exit conditions from handler (POLL only):
2680            **      Carry set=> Must be MEMERR
2681            **      HEX mode.
2682            **      C[0-3] = Error number.
2683            **
2684            ** Available subroutine levels:
2685            **      7
2686            **
2687            **
2688            ** NOTE:
2689            **      If handled or not, S11 & D1 must be preserved
2690            **
2691            ** What registers/RAM may be used if handled?:
2692            **      A-D,D0,R0-R3,S6
2693            **
2694            ** What registers/RAM may be used if not handled?:
2695            **      B,C,D,D0,R0-R3,S6
2696            **
2697            ** What registers/RAM may be used if error exit (POLL only)?:
2698            **      N/A
2699            **
2700            ** Special memory/pointer considerations (are pointers funny?):
2701            **      N/A
2702            **
2703            ** Envisioned application(s):
2704            **      To designate non-BASIC file as current, so cursor keys
2705            **      could be used to 'scroll' through the file contents.
2706            **
2707            **      Also possibly to be used in conjunction with the parse
2708            **      take-over poll, ie position at a non-BASIC file and
2709            **      enter lines.
```

```
2710                **
2711                ** History:
2712                **
2713                **    Date     Programmer          Modification
2714                **  --------   ----------  ----------------------------------
2715                **  03/04/83   S.W.        Added poll
2716                **
2717                ****************************************************************
2718                ****************************************************************
2719                *
2720 0A4E6 137   =EDIT20 CD1EX                  Save new CURRST
2721 0A4E9 06           RSTK=C
2722 0A4EB 8E00         GOSUBL =CLPSTK
           00
2723 0A4F1 07           C=RSTK
2724 0A4F3 1BD5  EDIT30 DO=(5) =CURRST
           5F2
2725 0A4FA 144          DATO=C A                UPDATE CURRST
2726 0A4FD D5           B=C    A                SAVE CURRST
2727                *
2728                * Initialize CURRL RAM location
2729                *
2730                * Note:
2731                *   EDIT20 is a RUN entry point
2732                *   Added RUN Binary code to JP&SYS/RUN
2733                *   Assumes: CURRST/CURREN can be non-BASIC
2734                *            All BASIC-file-dependent code must do their own
2735                *             check for the BASIC file type
2736                *
2737 0A4FF 21           P=     1                Avoid file type check
2738 0A501 8E00         GOSUBL =NULLP
           00
2739 0A507 D2           C=0    A
2740 0A509 490          GOC    EDIT45
2741 0A50C 171          D1=D1+ 2
2742 0A50F 15F3         C=DAT1 4                READ IN 1ST LINE# IN PROGRAM
2743 0A513 8E00  EDIT45 GOSUBL =SAVELO          WRITE OUT LINE# TO CURRL
           00
2744 0A519 1BC6         DO=(5) =CURREN
           5F2
2745 0A520 DB           C=D    A                D(A) SET IN NULLP
2746 0A522 144          DATO=C A
2747 0A525 03           RTNCC
2748                *
2749                *
2750 0A527 0000         REL(5) =EDITP
           0
2751                *
2752                * Assume S10,S11 clear on entry
2753                *
2754 0A52C 71DB  =EDIT  GOSUB  eolxc+
2755 0A530 5F0         GONC   EDIT70           NOT EOL => MUST BE FILENAME
2756                * Assume S10, S11 set as appropriate
2757                * No file specified - EDIT <workfile>
2758                * If workfile doesn't exist, create it
```

```
2759 0A533 7CA0 =EDITWF GOSUB  WRKFIL
2760 0A537 AFA          A=C    W
2761 0A53A AC3          D=0    S
2762 0A53D 590          GONC   EDIT75         (B.E.T.)
2763               *
2764 0A540 79E9  EDIT70 GOSUB  FSPECx
2765 0A544 492          GOC    EDIT79         CARRY => ERR LOADED
2766               *
2767               * Assumes neither FINDF & CREATE use S10, S11
2768 0A547 AFB   EDIT75 C=D    W
2769               * If file needs to be created, it is created in the mainframe if:
2770               *  1) No location was specified OR 2) MAIN was specified
2771 0A54A 10B          R3=C                  Save INFO on where to search
2772 0A54D 721A         GOSUB  FINDF+
2773 0A551 535          GONC   EDIT80
2774 0A554 866          ?ST=0  6
2775 0A557 97           GOYES  EDIT83
2776 0A559 11B          C=R3
2777 0A55C AF7          D=C    W              RESTORE INFO ON WHERE TO CREATE FIL
2778               *
2779               * NO MATCH => CREATE FILE - MUST CHECK TO SEE IF IT'S IN ROM/RAM,
2780               *
2781 0A55F 102          R2=A
2782 0A562 D2           C=0    A
2783 0A564 3113         LC(2)  (oFLENh)+(oBSsod) total BASIC file hdr length
2784 0A568 8E00         GOSUBL =CRETF+
           00
2785 0A56E 416   EDIT79 GOC    EDIT83         CARRY => ERRORM LOADED
2786 0A571 111          A=R1
2787 0A574 131          D1=A                    START OF FILE HEADER
2788 0A577 11A          C=R2
2789               *
2790 0A57A 15DF         DAT1=C 1FNAMh         WRITE OUT FILE NAME
2791 0A57E 17F          D1=D1+ 1FNAMh         STEP OVER IT
2792 0A581 3341         LC(4)  =fBASIC
           2E
2793 0A587 15D3         DAT1=C 1FTYPh
2794 0A58B 17F          D1=D1+ (1FTYPh)+(1TIMEh)+(1DATEh)+(1FLAGh)
2795 0A58E 174          D1=D1+ 1FLENh         STEP OVER LENGTH
2796 0A591 AF2          C=0    W
2797 0A594 15D9         DAT1=C 10             ZERO OUT SUB & LBL CHAIN HEADERS
2798 0A598 179          D1=D1+ 10
2799 0A59B 3100         LC(2)  =tEOL
2800 0A59F 14D          DAT1=C B              EOL
2801 0A5A2 131          D1=A
2802               *
2803               * D1 @ FILE HEADER
2804               *
2805 0A5A5 8E00 =EDIT80 GOSUBL =BASKEY
           00
2806 0A5AB 876          ?ST=1  6              NOT BASIC?
2807 0A5AE E0           GOYES  EDTEXT         Poll on EDIT
2808 0A5B0 86A          ?ST=0  10             NOT CALLED BY PURGE?
2809 0A5B3 12           GOYES  EDIT85
2810               * CURR FILE PURGED - DON'T WANT TO Collapse stks or CATalog
```

```
2811 0A5B5 137   EDIT81 CD1EX
2812 0A5B8 6A3F         GOTO   EDIT30
2813            *
2814            * Non BASIC filetype --- POLL
2815            *
2816 0A5BC 739E  EDTEXT GOSUB  poll
2817 0A5C0 B2           CON(2) =pEDIT
2818 0A5C2 4D0          GOC    EDIT83          Errored? (Must be MEMERR)
2819 0A5C5 831          ?XM=0
2820 0A5C8 C0           GOYES  EDIT85          Handled and Okay ?
2821 0A5CA 3300  EDITXE LC(4)  =eFTYPE         No response --- Error
          00
2822 0A5D0 68CA  EDIT83 GOTO   bserr           GOLONG BSERR
2823            *
2824            *
2825            *
2826 0A5D4 7E0F  EDIT85 GOSUB  EDIT20          UPDATE CURRST & CURREN
2827 0A5D8 87B          ?ST=1  11              Don't want CATalog
2828 0A5DB 00           RTNYES
2829 0A5DD 8C00         GOLONG =CATEDT
          00
2830            *
2831 0A5E3 3F77 =WRKFIL LCASC  \elifkrow\
          F627
          B666
          96C6
          56
2832 0A5F5 03           RTNCC
2833            *
2834            *
2835            ****************************************************************
2836            ****************************************************************
2837            **
2838            ** Name:(S) RAMROM  -  Classify Memory Device
2839            **
2840            ** Category:   FILUTL
2841            **
2842            ** Purpose:  Returns info on whether file in RAM,IRAM,other
2843            **
2844            ** Entry:    D(S) preserved from FINDF call:
2845            **               =0 => Mainframe RAM
2846            **               =1 => IRAM
2847            **               =2 => ROM
2848            **               =3 => EEPROM
2849            **
2850            ** Exit:     CARRY SET => RAM
2851            **                        S8=1 => IN MAIN
2852            **                           0 => IRAM
2853            **                 CLR => non-RAM memory device
2854            **                        S8=0
2855            **
2856            ** Calls:    none
2857            **
2858            ** Uses:     S8, C(S)
2859            **
```

```
2860                ** Stack lvls: 0
2861                **
2862                ** History:
2863                **
2864                **    Date      Programmer   Modifications
2865                **  --------    ----------   -------------
2866                **  06/30/82    S.W.         Added documentation
2867                **  12/17/82    S.W.         Eliminated distinction
2868                **                            between ROM & other
2869                **                            non-RAM memory devices
2870                **
2871                ********************************************************************
2872                ********************************************************************
2873                *
2874 0A5F7 858    =RAMROM ST=1   8
2875 0A5FA ACB            C=D    S
2876 0A5FD A4E            C=C-1  S
2877 0A600 400            RTNC
2878 0A603 848            ST=0   8
2879 0A606 A4E            C=C-1  S
2880 0A609 01             RTN                    Carry set IFF IRAM
2881                *
2882                *
2883                ********************************************************************
2884                ********************************************************************
2885                **
2886                ** Name:(S) LOCADR  -  Locate, Classify Address's Memory Device
2887                ** Name:(S) CURDVC  -  Classify Current File's Device
2888                **
2889                ** Category:   FILUTL
2890                **
2891                ** Purpose:  Given a file address, returns information
2892                **           regarding the medium (MAIN,IRAM,ROM, etc.)
2893                **
2894                **           CURDVC entry assumes the file address is (CURRST).
2895                **
2896                ** Entry:    2 entry points:
2897                **             1) CURDVC - No additional requirements.
2898                **             2) LOCADR - C(A) = some address in the file
2899                **
2900                ** Exit:   Specified address in R2
2901                **         Carry clr => Legitimate address
2902                **                      D(S)=0 => MAIN
2903                **                          #0 => PORT
2904                **                                D(S) reflects memory type
2905                **                                    =1 => RAM
2906                **                                    =2 => ROM
2907                **                                    =3 => EEPROM
2908                **                                D(0)= Extender#
2909                **                                D(1)= Port#
2910                **                                D(7-2)=Rest of Config. entry
2911                **                      A(A)=D1=R2(A)
2912                **
2913                **         Carry set (LOCADR entry only) =>
2914                **                      Not a legitimate address
```

```
2915                    **
2916                    ** Calls:    ROMCHK, ROMFND, EOFLCH, D1=CRS
2917                    **
2918                    ** Stk lvls: 2
2919                    **
2920                    ** Uses:     A-D, D1, R1 & R2
2921                    **
2922                    ** Detail:   THE ADDRESS MUST BE WITHIN A FILE CHAIN, OR CARRY
2923                    **           WILL AUTOMATICALLY COME BACK SET.
2924                    **
2925                    ** History:
2926                    **
2927                    **    Date     Programmer   Modifications
2928                    **  --------   ----------   ------------
2929                    **  06/30/82   S.W.         Added documentation
2930                    **
2931                    ************************************************************
2932                    ************************************************************
2933                    *
2934                    *
2935 0A60B 8E00 =CURDVC GOSUBL =D1=CRS        Set C(A) to CURRST
           00
2936                    *
2937 0A611 AC3  =LOCADR D=0    S             MAIN
2938 0A614 1F17         D1=(5) =MAINEN
           5F2
2939 0A61B 143          A=DAT1 A
2940 0A61E 10A          R2=C
2941 0A621 8B6          ?C<A   A             IN MAIN?
2942 0A624 72           GOYES  LOCA25
2943                    *
2944 0A626 7D1E         GOSUB  romchk
2945 0A62A 400   LOCA10 RTNC                 NO [MORE] PORTS?
2946                    * NOT IN MAIN - CHECK PORTS - C POINTS TO START OF FILE CHAIN
2947 0A62D 112          A=R2                 SPECIFIED ADDRESS
2948 0A630 8B2          ?A<C   A             NOT ON THIS PORT?
2949 0A633 01           GOYES  LOCA20        SEE IF ANY MORE PORTS
2950                    * SCAN TO END OF FILE CHAIN
2951 0A635 8E00         GOSUBL =EOFLC1       CHECK FOR NULL CHAIN
           00
2952                    * C(A) POINTS TO END OF FILE CHAIN
2953 0A63B 112          A=R2
2954 0A63E 8BA          ?A<=C  A
2955 0A641 C0           GOYES  LOCA30
2956                    * SEE IF ANY MORE PORTS
2957 0A643 760E  LOCA20 GOSUB  romfnd
2958 0A647 62EF         GOTO   LOCA10        FOUND ANOTHER PORT?
2959                    *
2960 0A64B DA    LOCA25 A=C    A
2961 0A64D 131   LOCA30 D1=A
2962 0A650 03           RTNCC
2963                    *
2964                    *
```

```
2965                    STITLE Reference Adjust (RFADJ-)
2966            ********************************************************************
2967            ********************************************************************
2968            **
2969            ** Name:(S) RFAD-I  -  Adjust Refs when mem moves to lower addr
2970            ** Name:(S) RFAD--  -  Adjust Refs when mem moves to lower addr
2971            **
2972            ** Category: PTRUTL
2973            **
2974            ** Purpose:  Adjusts address references on the FOR/NEXT & GOSUB
2975            **           stacks, in FIBs, as well as RAM pointers
2976            **           (PCADDR -> IMRAD3) & (CURRST -> AVMEMS), when
2977            **           appropriate; this is to be used when part of
2978            **           program memory moves to lower address space
2979            **           (hence a negative offset will be added to the
2980            **           references)
2981            **
2982            **           RFAD-- entry is used to adjust pointers when the
2983            **           file chain in MAIN has moved.
2984            **
2985            **           RFAD-I entry is used to adjust pointers when a
2986            **           file chain in an IRAM has moved.
2987            **
2988            ** Entry:
2989            **           B(A) = Bgn destination - Bgn source  (offset)
2990            **           R0 contains Begin Source
2991            **           2 entry points:
2992            **                1) RFAD-- - End Source assumed to be (AVMEMS).
2993            **                2) RFAD-I - D1 points to a 5-nibble location
2994            **                            containing the address of the file
2995            **                            chain end.
2996            **
2997            ** Exit:     B(A)=offset
2998            **           R0=Bgn Source
2999            **           R1=Bgn Destination
3000            **           Carry Clear
3001            **           RFAD-I entry point  - D1 preserved
3002            **           All other entry pts - D1 pts to AVMEMS ram loc.
3003            **
3004            **
3005            ** Calls:    RFUPD-, RFAD58
3006            **           LXFND, CSRC10, CSLC5, FORUPD, RFAD97, BUFFIB,
3007            **           PRVADR, I/OFND, RFUPD+, RFAD86
3008            **
3009            ** Uses:     A, C, D, R0, R1, D0, D1
3010            **
3011            ** Stack lvls: 2 (PCUPDT)
3012            **
3013            ** Detail:   Zeroes out references on the GOSUB & FOR-NEXT
3014            **           stacks which point into purged address space.
3015            **
3016            ** Note:     Memory must be moved BEFORE calling this routine!
3017            **
3018            ** History:
3019            **
```

```
3020            **    Date     Programmer   Modifications
3021            **  --------   ----------   -------------
3022            **  07/01/82   S.W.         Added documentation
3023            **  12/29/82   S.W.         Updates CURRST -> AVMEMS
3024            **
3025            ****************************************************************
3026            ****************************************************************
3027            *
3028            *
3029 0A652 1F49 =RFAD-- D1=(5) =AVMEMS
           5F2
3030 0A659 110  =RFAD-I A=R0
3031 0A65C C0          A=A+B  A              COMPUTE BEGIN DEST.
3032 0A65E 101         R1=A
3033 0A661 AC3         D=0    S
3034 0A664 B47         D=D+1  S
3035 0A667 6D90        GOTO   PCUPD+         Update/Zero affected references
3036            *
3037            *
3038            ****************************************************************
3039            ****************************************************************
3040            **
3041            ** Name:(S) RFUPD+  -  Updates a ptr when mem moves
3042            **
3043            ** Category:   PTRUTL
3044            **
3045            ** Purpose:
3046            **      Adds offset to given address reference, if memory
3047            **      movement to lower address space calls for such adjust-
3048            **      ment.  Indicates if reference points to a part of
3049            **      memory that has just been purged.
3050            **
3051            ** Entry:
3052            **      D(S)=0 => memory expansion, else memory contraction
3053            **      R0=Bgn Source for MOVEUM
3054            **      R1=Bgn Destination for MOVEUM
3055            **      D0 points to RAM location containing address to
3056            **          check/update
3057            **      D1 points to Ram location containing ptr to end source
3058            **      B(A)=offset  (bgn destination)-(bgn source)
3059            **             This number will be negative!
3060            **
3061            ** Exit:
3062            **      B, D, R0-R3, D0 & D1 are as they were upon entry
3063            **      Carry set=> Reference into purged address space.
3064            **                  A(A)=Bgn Destination
3065            **            clr=> Reference has been updated if needed.
3066            **                  Correct reference in C(A) & in RAM pointed
3067            **                  to by D0.
3068            **
3069            **
3070            ** Calls:      none
3071            **
3072            ** Uses.......
3073            **  Inclusive: A(A), C(A)
```

```
3074                **
3075                ** Stk lvls:   0
3076                **
3077                ** History:
3078                **
3079                **    Date     Programmer            Modification
3080                **  --------   ----------   ---------------------------------
3081                **  07/01/82   S.W.         Added documentation
3082                **
3083                ********************************************************************
3084                ********************************************************************
3085                *
3086 0A66B 134      RFUP++ D0=C
3087                *
3088 0A66E 110      =RFUPD+ A=R0                  BEGIN SOURCE
3089 0A671 146             C=DAT0 A
3090 0A674 8B6             ?C<A   A              ADDRESS<BEGIN SOURCE?
3091 0A677 11              GOYES  RFUP-5
3092                *
3093                *  MUST CHECK HERE TO SEE WHETHER ADDRESS IN C IS IN FILE CHAIN
3094                *  Address >= Begin source
3095                *
3096 0A679 143             A=DAT1 A
3097 0A67C 8B2             ?C>A   A              ADDR GREATER THAN END SOURCE?
3098 0A67F 70              GOYES  RFUP-3
3099 0A681 C9              C=C+B  A              ADD OFFSET
3100 0A683 144             DAT0=C A              & UPDATE
3101 0A686 03       RFUP-3 RTNCC
3102                *
3103 0A688 94B      RFUP-5 ?D=0   S              Memory expansion?
3104 0A68B BF              GOYES  RFUP-3
3105 0A68D 111             A=R1                  BEGIN DEST.
3106 0A690 8BA             ?C>=A  A              ADDRESS>=BEGIN DEST?
3107 0A693 00              RTNYES                ADDR BETWEEN BGN SOURCE & BGN DEST
3108 0A695 01              RTN
3109                *
3110                *
3111                *
3112                ********************************************************************
3113                ********************************************************************
3114                **
3115                ** Name:    RFAD85 - Adjusts ptrs when mem moves=>higher addr
3116                **
3117                ** Category:   LOCAL
3118                **
3119                ** Purpose:
3120                **      Adjusts specified # of RAM pointers, to account for
3121                **      memory movement to higher address space
3122                **
3123                ** Entry:
3124                **   RFAD86:
3125                **      D0 positioned at first RAM pointer to update
3126                **      P= (16-n) pointers to check/adjust, e.g. if only
3127                **                one pointer to be checked P=15
3128                **      A(A)=R0 Begin Source for Moving memory down
```

```
3129                **      D1 positioned at pointer to end source for moving memory
3130                **              down
3131                **      B(A) (End dest)-(End Source)= offset to add to affected
3132                **              pointers - This will be a positive number!
3133                **   RFAD84: SAME AS ABOVE EXCEPT WILL EXPLICITLY SET P, D0 TO
3134                **           UPDATE CURRST-AVMEMS
3135                **
3136                ** Exit:
3137                **      Carry set
3138                **      P=0
3139                **      D0 5 nibbles past last RAM location updated
3140                **
3141                ** Calls:      RFUPD+
3142                **
3143                ** Uses.......
3144                **  Exclusive: P, D0
3145                **  Inclusive: C(A), A(A), P, D0
3146                **
3147                ** Stk lvls:   1
3148                **
3149                ** Detail:     Does not zero any references
3150                **
3151                ** History:
3152                **
3153                **    Date     Programmer              Modification
3154                **  --------   ----------   ----------------------------------
3155                **  07/04/82   S.W.         Added documentation
3156                **
3157                ********************************************************************
3158                ********************************************************************
3159                *
3160                * Update CURRST -> AVMEMS
3161                *
3162 0A697 1BD5  RFAD84 D0=(5) =UPD1ST
           5F2
3163 0A69E 23           P=     15-((UPD1EN)-(UPD1ST))/5
3164                *
3165 0A6A0 7ACF  RFAD86 GOSUB  RFUPD+
3166 0A6A4 164          D0=D0+ 5
3167 0A6A7 0C           P=P+1
3168 0A6A9 56F          GONC   RFAD86
3169 0A6AC 01           RTN
3170                *
3171                ********************************************************************
3172                ********************************************************************
3173                **
3174                ** Name:(S) FORUPD  -  FOR Stack Update
3175                **
3176                ** Category:   PTRUTL
3177                **
3178                ** Purpose:
3179                **      Updates references on FOR-NEXT stack
3180                **
3181                ** Entry:
3182                **      P      =  0
```

```
3183                **      R0 contains Begin Source
3184                **      D1 points to location, containing End Source
3185                **      If want appropriate references zeroed
3186                **         have D(S)#0 and R1 containing Begin Destination
3187                **      B(A) containing offset (Bgn Source)-(Bgn Dest)
3188                **
3189                ** Exit:
3190                **      P       =  0
3191                **
3192                ** Calls:       RFUP++
3193                **
3194                ** Uses.......
3195                **  Inclusive: A(A), C(A), D(A), D0
3196                **
3197                ** Stk lvls:   1
3198                **
3199                ** History:
3200                **
3201                **    Date      Programmer              Modification
3202                **  --------    ----------   ------------------------------------
3203                **  01/28/83    S.W.         Added routine
3204                **
3205                ********************************************************************
3206                ********************************************************************
3207                *
3208 0A6AE 1BE9 =FORUPD DO=(5) =FORSTK
           5F2
3209 0A6B5 146          C=DAT0 A
3210 0A6B8 D7           D=C    A
3211 0A6BA 164          DO=DO+ 5
3212 0A6BD 146          C=DAT0 A
3213 0A6C0 DF           CDEX   A
3214 0A6C2 8BF   FORUP2 ?C>=D  A
3215 0A6C5 00           RTNYES
3216 0A6C7 70AF         GOSUB  RFUP++
3217 0A6CB D2           C=0    A
3218 0A6CD 550          GONC   FORUP3          Not pting to purged space?
3219 0A6D0 144          DAT0=C A               Zero out reference
3220 0A6D3 3192  FORUP3 LC(2)  41
3221 0A6D7 132          ADOEX
3222 0A6DA C2           C=C+A  A
3223 0A6DC 132          ADOEX
3224 0A6DF 52E          GONC   FORUP2          (B.E.T.)
3225                *
3226                ********************************************************************
3227                ********************************************************************
3228                **
3229                ** Name:    RFAD96  -  Adjusts ptrs if mem moves=>lower addr
3230                **
3231                ** Category:   LOCAL
3232                **
3233                ** Purpose:
3234                **      Adjusts specified # of RAM pointers, to account for
3235                **      file memory movement to lower address space
3236                **
```

```
3237            ** Entry:
3238            **      D(S)#0 => references pointing into purged address
3239            **                space will be zeroed
3240            **      D0 positioned at first RAM pointer to update
3241            **      P= (16-n) pointers to check/adjust, e.g. if only
3242            **         one pointer to be checked, P=15
3243            **      A(A)=R0 Begin source for moving memory up
3244            **      D1 positioned at pointer to end source
3245            **      B(A) offset (Bgn dest)-(Bgn source)   NEGATIVE!
3246            **
3247            ** Exit:
3248            **      P      =  0
3249            **      Carry set
3250            **      D0 5 nibbles past last RAM location checked
3251            **      Any pointer pointing into purged address space
3252            **       is zeroed
3253            **
3254            ** Calls: RFUPD-
3255            **
3256            ** Uses:  A(A), C(A), P, D0
3257            **
3258            ** Stk lvls:   1
3259            **
3260            ** History:
3261            **
3262            **    Date     Programmer              Modification
3263            **  --------   ----------   ------------------------------------
3264            **  11/04/82   S.W.         Wrote documentation
3265            **
3266            ***************************************************************
3267            ***************************************************************
3268            *
3269            * Carry must be set on return for KEYDEL
3270            *
3271 0A6E2 788F  RFAD97 GOSUB  RFUPD+
3272 0A6E6 570         GONC   RFAD98
3273            * Only necessary for RFADJ-
3274 0A6E9 D2          C=0    A
3275 0A6EB 144         DAT0=C A
3276            *
3277 0A6EE 164   RFAD98 D0=D0+ 5
3278 0A6F1 0C          P=P+1
3279 0A6F3 5EE         GONC   RFAD97
3280 0A6F6 01          RTN
3281            *
3282            *
3283            *
3284            ***************************************************************
3285            ***************************************************************
3286            **
3287            ** Name:    RFADJ+  -  Adjusts Refs When Mem Moves=>Higher Addr
3288            ** Name:(S) RFAD+,  -  Adjusts Refs When Mem Moves=>Higher Addr
3289            ** Name:(S) RFAD+I  -  Adjusts Refs When Mem Moves=>Higher Addr
3290            **
3291            ** Category: PTRUTL
```

```
3292                **
3293                ** Purpose:  Adjust address references on the FOR/NEXT & GOSUB
3294                **           stacks, in the FIBs, as well as the RAM locations
3295                **           PCADDR -> TMRAD3 & CURRST -> AVMEMS, to reflect
3296                **           instances of program memory expanding into
3297                **           higher address space.
3298                **
3299                ** Entry:
3300                **           B(A)=   Offset  (End Dest.)-(End Source)
3301                **                      This number will be positive!
3302                **           3 entry points:
3303                **                1) RFADJ+  - Bgn source in A(A).
3304                **                2) RFAD++  - Bgn source already in R0.
3305                **                3) RFAD+I  - D1 pointing to RAM location
3306                **                             containing pointer to end of
3307                **                             file chain - entry pt for IRAMS.
3308                **                             Bgn source already in R0.
3309                **
3310                ** Exit:     B(A)=OFFSET; R0=BGN SOURCE; CARRY CLEAR
3311                **           C(S )=0 => Some address on GOSUB or FOR-NEXT
3312                **                      referenced block that moved
3313                **
3314                ** Calls:    RFAD58, RFUPD+, RFAD85
3315                **           RFAD97, BUFFIB, LXFND, CSRC10, CSLC5, FORUPD,
3316                **           I/OFND, PRVADR, RFAD86
3317                **
3318                ** Uses:     A, C, D, D0, D1, R0
3319                **
3320                ** Stk lvls: 2 (PCUPDT)
3321                **
3322                ** Detail:  Needed when program mem moves to higher address space
3323                **
3324                ** Note:    Memory must be moved BEFORE calling this routine!
3325                **
3326                ** History:
3327                **
3328                **    Date     Programmer   Modifications
3329                **  --------   ----------   -------------
3330                **  07/01/82   S.W.         Added documentation
3331                **  12/29/82   S.W.         Updates CURRST -> AVMEMS
3332                **
3333                ****************************************************************
3334                ****************************************************************
3335                *
3336                *
3337 0A6F8 100    =RFADJ+ R0=A
3338 0A6FB 1F49 =RFAD++ D1=(5) =AVMEMS
           5F2
3339 0A702 AC3    =RFAD+I D=0    S
3340                *
3341                * Starting from current FORSTK to RAMEND make one pass to
3342                * adjust all the addresses on the stack
3343                *
3344                * Set D(14,10) = CALSTK
3345                *     D(9,5)   = ACTIVE
```

```
3346                *
3347 0A705 1BDA    PCUPD+ D0=(5) =CALSTK
           5F2
3348 0A70C 146             C=DAT0 A
3349 0A70F 7112            GOSUB  Cslc5
3350 0A713 184             D0=D0- (CALSTK)-(ACTIVE)
3351 0A716 146             C=DAT0 A
3352 0A719 7702            GOSUB  Cslc5
3353 0A71D AFF             CDEX   W
3354 0A720 AC7             D=C    S
3355                *
3356                * Now adjust the FOR-NEXT stack
3357                *
3358 0A723 778F            GOSUB  FORUPD
3359                *
3360                * Set up to adjust GSBSTK
3361                *   Set D0   = D(A)
3362                *       D(A) = D(9,5)
3363                *
3364 0A727 DB      RFAD10  C=D    A
3365 0A729 134             D0=C
3366 0A72C AFB             C=D    W
3367 0A72F 8E00            GOSUBL =csrc5
           00
3368 0A735 D7              D=C    A                D(A) = Current ACTIVE
3369                *
3370                * Adjust the GSBSTK
3371                *
3372 0A737 136     RFAD15  CDOEX
3373 0A73A 8BF             ?C>=D  A                Reached ACTIVE ?
3374 0A73D F4              GOYES  RFAD30           If so, done with current program
3375 0A73F 134             D0=C
3376 0A742 15A0            A=DAT0 1
3377 0A746 160             D0=D0+ 1
3378 0A749 B04             A=A+1  P
3379 0A74C 5A0             GONC   RFAD18           NOt a BOUNDARY or MICROCODE rtn ?
3380 0A74F 142             A=DAT0 A
3381 0A752 8A8             ?A=0   A                A BOUNDARY ?
3382 0A755 B0              GOYES  RFAD20
3383                *
3384 0A757 2F      RFAD18  P=     15               Want RTN
3385 0A759 758F            GOSUB  RFAD97           Zero out appropriate addresses
3386 0A75D 49D             GOC    RFAD15           (B.E.T.)
3387                *
3388                * Reached F00000, must a User-Defined Function block follows.
3389                * Adjust all the addresses saved by User-Defined Function
3390                *
3391 0A760 164     RFAD20  D0=D0+ 5                Skip over the 00000
3392 0A763 29              P=     9
3393 0A765 773F            GOSUB  RFAD86
3394 0A769 16F             D0=D0+ 16
3395 0A76C 16E             D0=D0+ 15               D0 @ saved S-R1-2
3396 0A76F 7BFE            GOSUB  RFUPD+
3397 0A773 16A             D0=D0+ 6+5              D0 @ offset of previous FORSTK
3398                *
```

```
3399                * Now DO is pointing at the offset of the previous FORSTK
3400                * Set DO   = Previous GSBSTK
3401                *     D(A) = Previous FORSTK
3402                *
3403 0A776 7D91  RFAD25 GOSUB  PRVADR
3404 0A77A D7          D=C    A              D(A) = Previous FORSTK
3405 0A77C 164         DO=DO+ 5
3406 0A77F 7491        GOSUB  PRVADR
3407 0A783 DF          CDEX   A              D(A) = Previous GSBSTK
3408 0A785 793F        GOSUB  FORUP2         C(A) = previous FORSTK
3409 0A789 4D9         GOC    RFAD10         (B.E.T.)
3410                *
3411                * Done with current program, see if still in a subprogram ?
3412                * Check if D(14,10) = RAMEND ?
3413                *
3414 0A78C AFB   RFAD30 C=D    W
3415 0A78F 8F00        GOSBVL =CSRC10
                    000
3416 0A796 1B2B        DO=(5) =RAMEND
                    5F2
3417 0A79D 142         A=DAT0 A
3418 0A7A0 8A2         ?A=C   A              If current CALSTK = RAMEND
3419 0A7A3 97          GOYES  RFAD75          then done
3420                *
3421                * Adjust addresses saved by CALL
3422                *
3423 0A7A5 AC2         C=0    S              Initialize mainframe block flag
3424                *
3425 0A7A8 134   RFAD35 DO=C                 DO @ Current CALSTK
3426 0A7AB 146         C=DAT0 A
3427 0A7AE 96E         ?C#0   B              Block ID # 00
3428 0A7B1 50          GOYES  RFAD40         Mainframe's block ID is 00
3429 0A7B3 B46         C=C+1  S              Use C(S) to flag end of save blocks
3430                *
3431                * Now compute the address of next block if any
3432                *
3433 0A7B6 D7    RFAD40 D=C    A
3434 0A7B8 164         DO=DO+ 5
3435 0A7BB F7          DSR    A
3436 0A7BD F7          DSR    A              D(A) = Current block size
3437 0A7BF 136         CDOEX
3438 0A7C2 134         DO=C
3439 0A7C5 C3          D=D+C  A              Save next block address in D(A)
3440 0A7C7 14E         C=DAT0 B
3441 0A7CA 90A         ?C=0   P              Any addr to adjust in this block ?
3442 0A7CD 52          GOYES  RFAD50         If not, goto next block if any
3443 0A7CF 160         DO=DO+ 1              Skip over the address count
3444 0A7D2 B8A         C=-C   P              Complement the count
3445 0A7D5 80F0        CPEX   0
3446 0A7D9 719E  RFAD45 GOSUB  RFUPD+
3447 0A7DD 5C0         GONC   RFAD46
3448                * Don't zero the address if the addr = Beg. Dest.
3449 0A7E0 8A2         ?A=C   A
3450 0A7E3 70          GOYES  RFAD46
3451 0A7E5 D2          C=0    A
```

```
3452 0A7E7 144          DAT0=C A
3453               *
3454 0A7EA 164   RFAD46 DO=DO+ 5
3455 0A7ED 0C           P=P+1
3456 0A7EF 59E          GONC   RFAD45
3457               *
3458 0A7F2 DB    RFAD50 C=D    A
3459 0A7F4 94A          ?C=0   S              Done with mainframe block yet ?
3460 0A7F7 1B           GOYES  RFAD35         If not, keep going
3461               *
3462               * Set up for previous program environment
3463               * set D(14,10) = previous CALSTK
3464               *     D(9,5)   = Previous TACTVE
3465               *
3466 0A7F9 16E          DO=DO+ 15             DO @ offset to previous CALSTK
3467 0A7FC 7711         GOSUB  PRVADR
3468 0A800 7021         GOSUB  Cslc5
3469 0A804 184          DO=DO- 5              DO @ Offset to previous ACTIVE
3470 0A807 7C01         GOSUB  PRVADR
3471 0A80B 7511         GOSUB  Cslc5
3472 0A80F AFF          CDEX   W
3473 0A812 AC7          D=C    S
3474 0A815 189          DO=DO- 10             DO @ Offset to previous FORSTK
3475 0A818 6D5F         GOTO   RFAD25         Start from the FORSTK again
3476               *
3477               *  Now adjust other pointers in the system RAM
3478               *   DSPCHX -> TMRAD3
3479               *
3480 0A81C 25    RFAD75 P=     15-((UPD2EN)-(UPD2ST))/5
3481 0A81E 1B47         DO=(5) =UPD2ST
          6F2
3482 0A825 79BE         GOSUB  RFAD97         Zeroes pointers when appropriate
3483               *
3484               * Adjust CURRST -> AVMEMS
3485               *
3486 0A829 7A6E         GOSUB  RFAD84         None of these will be zeroed
3487               * Exits with DO at AVMEME
3488 0A82D 1917         DO=(2) =MAINEN
3489 0A831 146          C=DAT0 A              Update addresses in I/O Buffers
3490               * Update references in the I/O buffers
3491 0A834 134   EXTADR DO=C
3492 0A837 15E0         C=DAT0 1              Read in # of addresses to check
3493 0A83B B8A          C=-C   P              Negate for RFAD97
3494 0A83E 80D0         P=C    0              Complement # of addresses to update
3495 0A842 160          DO=DO+ 1              Point to buffer ID
3496 0A845 15A5         A=DAT0 6              Read in ID & Length
3497 0A849 938          ?A=0   X              No more buffers?
3498 0A84C 42           GOYES  PCUPDT         External Update Done
3499 0A84E BF4          ASR    W
3500 0A851 F4           ASR    A
3501 0A853 F4           ASR    A              A(A) contains length
3502 0A855 165          DO=DO+ 6              Step over header
3503 0A858 136          CDOEX
3504 0A85B 134          DO=C                  Restore DO in case of update
3505 0A85E C2           C=C+A  A              C(A) points past buffer
```

```
3506 0A860 890          ?P=    0              No pointers to update?
3507 0A863 1D           GOYES  EXTADR
3508 0A865 D7           D=C    A              Pointer to next buffer
3509 0A867 777E         GOSUB  RFAD97         Update pointers
3510 0A86B DB           C=D    A
3511 0A86D 46C          GOC    EXTADR         (B.E.T.) - Look at next buffer
3512                *
3513                * Update addresses in FIB when memory moved
3514                *
3515                *  USED A,C,D,D0 +2 SUB LEVEL
3516                *
3517 0A870 137   PCUPDT CD1EX                  SAVE D1 IN D(A)
3518 0A873 D7           D=C    A
3519 0A875 8E00         GOSUBL =BUFFIB
             00
3520 0A87B 7B80         GOSUB  D1=D
3521 0A87F 536          GONC   LXU100         go Update LEX Buffer
3522 0A882 110          A=R0
3523 0A885 14E   PCU310 C=DAT0 B
3524 0A888 96A          ?C=0   B              REACHED END OF FIB YET ?
3525 0A88B 85           GOYES  LXU100         go Update LEX Buffer
3526 0A88D 16B          D0=D0+ (oDEVCb)
3527 0A890 1560         C=DAT0 P              C(0) = DEVICE TYPE
3528 0A894 160          D0=D0+ (oFBEGb)-(oDEVCb)
3529 0A897 A0E          C=C-1  P              IS THIS A MAINFRAME FILE ?
3530 0A89A 4E0          GOC    PCU340         IF SO, TAKE A LOOK AT THE POINTERS
3531 0A89D A0E          C=C-1  P              IS THIS AN IRAM FILE ?
3532 0A8A0 480          GOC    PCU340         IF SO, CHECK THE ADDR TOO
3533 0A8A3 167          D0=D0+ (oDBEGb)-(oFBEGb)
3534 0A8A6 503          GONC   PCU350         (B.E.T.)
3535 0A8A9 94B   PCU340 ?D=0   S              Memory expand?
3536 0A8AC 51           GOYES  PCU342
3537 0A8AE 146          C=DAT0 A
3538 0A8B1 111          A=R1
3539 0A8B4 8A6          ?A#C   A              File Start#Bgn Dest?
3540 0A8B7 A0           GOYES  PCU342
3541 0A8B9 D2           C=0    A
3542 0A8BB 144          DAT0=C A
3543 0A8BE 560          GONC   PCU343         (B.E.T.)
3544                *
3545 0A8C1 79AD  PCU342 GOSUB  RFUPD+          UPDATE FILE HEADER ADDRESS
3546 0A8C5 167   PCU343 D0=D0+ (oDBEGb)-(oFBEGb)
3547 0A8C8 146          C=DAT0 A
3548 0A8CB 110          A=R0
3549                * If start of data = bgn source (empty file), don't update
3550 0A8CE 8A2          ?A=C   A
3551 0A8D1 60           GOYES  PCU350
3552 0A8D3 779D         GOSUB  RFUPD+          UPDATE DATA START ADDRESS
3553 0A8D7 16E   PCU350 D0=D0+ (oRECLb)-(oDBEGb)
3554 0A8DA 16F          D0=D0+ (oRLENb)-(oRECLb)
3555 0A8DD 16A          D0=D0+ (1FIB)-(oRLENb)
3556 0A8E0 54A          GONC   PCU310         (B.E.T.)
3557                *
3558                * Update Main Table Addresses in LEX Buffer
3559                *
```

```
3560 0A8E3 8E00  LXU100 GOSUBL =LXFND          Find LEX Buffer
           00
3561 0A8E9 7D10         GOSUB  D1=D            Restore D1 from D(A)
3562 0A8ED 500          RTNNC                  Buffer not found
3563 0A8F0 110          A=R0                   Restore address to check against
3564 0A8F3 14E   LXU110 C=DATO B               Read Lex ID
3565 0A8F6 96A          ?C=0   B               End of External Lex IDs?
3566 0A8F9 F0           GOYES  CLRCAR          DONE
3567 0A8FB 165          D0=D0+ (1LXID)+(1LXTKR) Skip Lex ID and Token range
3568 0A8FE 7C6D         GOSUB  RFUPD+          Update Main table address
3569 0A902 164          D0=D0+ 1LXADR          Skip Main Table address
3570 0A905 5DE          GONC   LXU110          B.E.T.
3571 0A908 03    CLRCAR RTNCC
3572           ********************************************************************
3573 0A90A 137   D1=D    CD1EX
3574 0A90D 134          D0=C
3575 0A910 DB           C=D    A
3576 0A912 135          D1=C
3577 0A915 01           RTN
3578           *
3579           *
3580 0A917 146   PRVADR C=DATO A
3581 0A91A 132          ADOEX
3582 0A91D 130          D0=A
3583 0A920 C2           C=C+A  A
3584 0A922 03           RTNCC
3585           *
3586 0A924 8D00 =Cslc5  GOVLNG =CSLC5
           000
3587           *
3588           *
3589           *
```

```
 ACTIVE  Abs  193960 #2F5A8 -    12  3350
 ARGERR  Ext               -   697
 ARGSTA  Ext               -   656
 AVE=D1  Ext               -   902
 AVM2DS  Ext               -  2538
 AVMEMS  Abs  193940 #2F594 -    12  1699  1775  3029  3338
 BASKEY  Ext               -  2016  2805
=BLKOK   Abs   40576 #09E80 -   585   325
 BLKOK2  Abs   40579 #09E83 -   586   297   535   581
 BLNKC+  Ext               -  2512
 BSERR   Ext               -  1450
 BUFFIB  Ext               -  3519
 Bserr   Abs   41746 #0A312 -  2043  2033
=C=MAIN  Abs   40976 #0A010 -  1213  1147  1464
 CALSTK  Abs  193965 #2F5AD -    12  3347  3350
 CAT$83  Ext               -  2527
 CAT$90  Ext               -  2535
 CATC++  Ext               -   377
 CATEDT  Ext               -  2829
 CHKSGN  Abs   40626 #09EB2 -   660   657
 CLPSTK  Ext               -  2722
 CLRCAR  Abs   43272 #0A908 -  3571  3566
 CLRFRC  Ext               -   671   678
 CNVWUC  Ext               -   304
 CRETF+  Ext               -  2784
 CRETM5  Ext               -  1777
 CSLC5   Ext               -  3586
 CSRC10  Ext               -  3415
=CURDVC  Abs   42507 #0A60B -  2935  1423  2343
 CURREN  Abs  193900 #2F56C -    12  2744
 CURRST  Abs  193885 #2F55D -    12  2724
=Cslc5   Abs   43300 #0A924 -  3586  3349  3352  3468  3471
 D1=CRS  Ext               -  1719  2935
 D1=D    Abs   43274 #0A90A -  3573  3520  3561
 DECHEX  Ext               -   556
 DIGCK+  Abs   40528 #09E50 -   569   516   527   530
 DIGCK2  Abs   40544 #09E60 -   573   525
 DIGCK3  Abs   40574 #09E7E -   583   580
 DRANGE  Ext               -   569
=DVCNFE  Abs   42073 #0A459 -  2500  2508
 DVCTYP  Abs   40293 #09D65 -   423   239   383
=EDIT    Abs   42284 #0A52C -  2754
=EDIT20  Abs   42214 #0A4E6 -  2720  2826
 EDIT30  Abs   42227 #0A4F3 -  2724  2812
 EDIT45  Abs   42259 #0A513 -  2743  2740
 EDIT70  Abs   42304 #0A540 -  2764  2755
 EDIT75  Abs   42311 #0A547 -  2768  2762
 EDIT79  Abs   42350 #0A56E -  2785  1778  2765
=EDIT80  Abs   42405 #0A5A5 -  2805  2773
 EDIT81  Abs   42421 #0A5B5 -  2811  1803
 EDIT83  Abs   42448 #0A5D0 -  2822  2775  2785  2818
 EDIT85  Abs   42452 #0A5D4 -  2826  2809  2820
 EDITP   Ext               -  2750
=EDITWF  Abs   42291 #0A533 -  2759
 EDITXE  Abs   42442 #0A5CA -  2821  2050  2369
```

```
 EDTEXT  Abs   42428 #0A5BC -  2816  2807
 ENDALL  Ext                -  1431
 EOFLC1  Ext                -  2951
 EOFLCH  Ext                -  1678
 EOLXCK  Ext                -  1494
 EOS     Abs   40590 #09E8E -   591   509   573   579
 EXPEX+  Ext                -   243   410   655
 EXTADR  Abs   43060 #0A834 -  3491  3507  3511
=FILEF   Abs   40880 #09FB0 -  1140  2112
 FILEF-  Abs   40899 #09FC3 -  1154  1202
 FILEP!  Ext                -   277
 FILEP+  Ext                -   372
 FILEXQ  Abs   40138 #09CCA -   368   438
 FILF05  Abs   40886 #09FB6 -  1147  1100
 FILF10  Abs   40896 #09FC0 -  1153  1168
 FILF30  Abs   40915 #09FD3 -  1166  1158
 FILF40  Abs   40927 #09FDF -  1175  1156
 FILF50  Abs   40943 #09FEF -  1196  1116
 FILF52  Abs   40947 #09FF3 -  1197  1211
 FILF55  Abs   40950 #09FF6 -  1198  1178
 FILF60  Abs   40956 #09FFC -  1199  1131
 FILF70  Abs   40968 #0A008 -  1210  1189
=FILFMF  Abs   40883 #09FB3 -  1141  1107
 FILSK+  Ext                -  1167
 FILSKP  Ext                -  1714
=FILXQ$  Abs   39829 #09B95 -   245
=FILXQ^  Abs   39798 #09B76 -   232   889  2105
 FINDA   Ext                -   426
=FINDF   Abs   40823 #09F77 -  1097  2003
=FINDF+  Abs   40803 #09F63 -  1084  1442  1988  2351  2772
 FINDW+  Abs   40877 #09FAD -  1136
=FINDWF  Abs   40873 #09FA9 -  1135  1727  1767  2116
 FLSKPB  Ext                -  1480
 FLXQ05  Abs   39886 #09BCE -   269   250
 FLXQ07  Abs   39888 #09BD0 -   271   265
 FLXQ10  Abs   39897 #09BD9 -   274   267
 FLXQ11  Abs   39937 #09C01 -   290   260
 FLXQ12  Abs   39940 #09C04 -   291   287
 FLXQ13  Abs   39974 #09C26 -   304   296
 FLXQ14  Abs   40021 #09C55 -   320   317
 FLXQ15  Abs   40038 #09C66 -   328   323
 FLXQ16  Abs   40086 #09C96 -   343   394
 FLXQ17  Abs   40089 #09C99 -   344   337   428
 FLXQ18  Abs   40097 #09CA1 -   347   401   421
 FLXQ19  Abs   40100 #09CA4 -   348   333   432
 FLXQ22  Abs   40103 #09CA7 -   349   302   386   392
 FLXQ25  Abs   40106 #09CAA -   350   288
 FLXQ29  Abs   40120 #09CB8 -   358   434
 FLXQ30  Abs   40136 #09CC8 -   364   361   378
 FLXQ40  Abs   40170 #09CEA -   380   375
 FLXQ65  Abs   40185 #09CF9 -   388   436
 FLXQ70  Abs   40209 #09D11 -   396   430
 FLXQ74  Abs   40224 #09D20 -   401
 FLXQ75  Abs   40228 #09D24 -   405   399
 FLXQ80  Abs   40286 #09D5E -   420   562
```

```
 FLXQOK  Abs   39997 #09C3D -   313   310
 FLXQPL  Abs   40122 #09CBA -   359   326   341
 FORSTK  Abs  193950 #2F59E -    12  3208
 FORUP2  Abs   42690 #0A6C2 -  3214  3224  3408
 FORUP3  Abs   42707 #0A6D3 -  3220  3218
=FORUPD  Abs   42670 #0A6AE -  3208  3358
 FRAC15  Ext                  687
 FSPC10  Abs   40767 #09F3F -   900   914
 FSPC15  Abs   40779 #09F4B -   903   901
 FSPC20  Abs   40781 #09F4D -   910   899
 FSPC30  Abs   40795 #09F5B -   915  1088  1090
=FSPCER  Abs   40446 #09DFE -   533   510   521   536   542   548   550   570
                             574  1458  2045
=FSPECx  Abs   40749 #09F2D -   889  1439  2349  2764
 FTYPFD  Ext                 2364
 GETPR1  Ext                 1476  1691  2359
 GTPR#1  Abs   40737 #09F21 -   699   695
 GTPRT#  Abs   40713 #09F09 -   691   680
 INVFSP  Ext                  533
=LAKEYS  Abs   41836 #0A36C -  2179  2008
 LEXBF+  Ext                 1756
 LOCA10  Abs   42538 #0A62A -  2945  2958
 LOCA20  Abs   42563 #0A643 -  2957  2949
 LOCA25  Abs   42571 #0A64B -  2960  2942
 LOCA30  Abs   42573 #0A64D -  2961  2955
=LOCADR  Abs   42513 #0A611 -  2937
 LXFND   Ext                 3560
 LXU100  Abs   43235 #0A8E3 -  3560  3521  3525
 LXU110  Abs   43251 #0A8F3 -  3564  3570
 MAINEN  Abs  193905 #2F571 -    12  2938  3488
 MAINST  Abs  193880 #2F558 -    12  1213
 MEMCKL  Ext                 1744
 MFERR   Ext                 2048
 MFWRQ8  Ext                 1447
 MOVEU1  Ext                 1749
 MSIZE+  Ext                 2524
=NAME    Abs   41778 #0A332 -  2105
 NAMEDC  Ext                 2100  2439
 NAMEP   Ext                 2101
 NULLP   Ext                 2738
 NXTSTM  Ext                 1429
 OUTC15  Ext                 2514
 OUTNBS  Ext                 2516
 PCU310  Abs   43141 #0A885 -  3523  3556
 PCU340  Abs   43177 #0A8A9 -  3535  3530  3532
 PCU342  Abs   43201 #0A8C1 -  3545  3536  3540
 PCU343  Abs   43205 #0A8C5 -  3546  3543
 PCU350  Abs   43223 #0A8D7 -  3553  3534  3551
 PCUPD+  Abs   42757 #0A705 -  3347  3035
 PCUPDT  Abs   43120 #0A870 -  3517  3498
=PDEV    Abs   40606 #09E9E -   655
=PDEV+   Abs   40603 #09E9B -   654
=PDEV1   Abs   40612 #09EA4 -   656   411
 PDEV3   Abs   40631 #09EB7 -   662   659
 PDVE24  Abs   40731 #09F1B -   697   661   667
```

```
 POLL    Ext                  -  2498
=POLlj   Abs   42067 #0A453  -  2497
 PRGEXT  Abs   41119 #0A09F  -  1452  1446
 PRGF    Abs   41317 #0A165  -  1699  1485
 PRGF05  Abs   41309 #0A15D  -  1695  1693
 PRGF20  Abs   41327 #0A16F  -  1701  1685
 PRGF25  Abs   41364 #0A194  -  1712  1709
 PRGF27  Abs   41431 #0A1D7  -  1744  1734
 PRGF35  Abs   41441 #0A1E1  -  1747  1721  1728  1737  1739
 PRGF37  Abs   41463 #0A1F7  -  1753  1673
 PRGF40  Abs   41474 #0A202  -  1757  1460  1754
 PRGF80  Abs   41552 #0A250  -  1790  1786  1804
 PRGF85  Abs   41560 #0A258  -  1794  1759  1791
 PRGF90  Abs   41576 #0A268  -  1803  1768
=PRGFE   Abs   41240 #0A118  -  1670  2042
=PRGFMF  Abs   41286 #0A146  -  1687  1424
 PRGIRM  Abs   41252 #0A124  -  1675  1696
 PRGRO+  Abs   41226 #0A10A  -  1665  1689
=PRGROM  Abs   41742 #0A30E  -  2042
 PRGRTN  Abs   41248 #0A120  -  1673  1669
=PRIVAT  Abs   42048 #0A440  -  2441
 PRIVTP  Ext                  -  2440
 PRT#DC  Ext                  -  2521
 PRT#N+  Abs   40452 #09E04  -   535   507
 PRT#NL  Abs   40459 #09E0B  -   537   502
 PRT#P   Abs   40346 #09D9A  -   492   318
 PRT#P5  Abs   40425 #09DE9  -   525   514
 PRT#P6  Abs   40467 #09E13  -   541   526
 PRT#P7  Abs   40472 #09E18  -   543   517
 PRT#P8  Abs   40475 #09E1B  -   544   528
 PRT#P9  Abs   40478 #09E1E  -   546   531
 PRVADR  Abs   43287 #0A917  -  3580  3403  3406  3467  3470
 PRVT20  Abs   41971 #0A3F3  -  2369  2365
 PUGFIB  Ext                  -  1757
 PURG00  Abs   41044 #0A054  -  1424  1443
 PURG10  Abs   41051 #0A05B  -  1426  1468
 PURG20  Abs   41068 #0A06C  -  1433  1421
 PURG25  Abs   41091 #0A083  -  1442  1434
 PURG30  Abs   41137 #0A0B1  -  1460  1456
 PURG40  Abs   41144 #0A0B8  -  1463  1437
 PURG45  Abs   41151 #0A0BF  -  1465  1481  1490
 PURG47  Abs   41194 #0A0EA  -  1484  1478
 PURGDC  Ext                  -  1418
 PURGDN  Abs   41056 #0A060  -  1429  1461  2022  2388  2542
=PURGE   Abs   41033 #0A049  -  1420
 PURGEP  Ext                  -  1419
 PURGNF  Abs   41056 #0A060  -  1428  1448
 RAMEND  Abs  193970 #2F5B2  -    12  3416
=RAMROM  Abs   42487 #0A5F7  -  2874  1688  1991  2353
 RDENTY  Ext                  -  2366
 RDINFO  Ext                  -  1980  1998
 REVPOP  Ext                  -   247
=RFAD++  Abs   42747 #0A6FB  -  3338
=RFAD+I  Abs   42754 #0A702  -  3339
=RFAD--  Abs   42578 #0A652  -  3029
```

```
=RFAD-I  Abs   42585 #0A659 -  3030  1752
 RFAD10  Abs   42791 #0A727 -  3364  3409
 RFAD15  Abs   42807 #0A737 -  3372  3386
 RFAD18  Abs   42839 #0A757 -  3384  3379
 RFAD20  Abs   42848 #0A760 -  3391  3382
 RFAD25  Abs   42870 #0A776 -  3403  3475
 RFAD30  Abs   42892 #0A78C -  3414  3374
 RFAD35  Abs   42920 #0A7A8 -  3425  3460
 RFAD40  Abs   42934 #0A7B6 -  3433  3428
 RFAD45  Abs   42969 #0A7D9 -  3446  3456
 RFAD46  Abs   42986 #0A7EA -  3454  3447  3450
 RFAD50  Abs   42994 #0A7F2 -  3458  3442
 RFAD75  Abs   43036 #0A81C -  3480  3419
 RFAD84  Abs   42647 #0A697 -  3162  3486
 RFAD86  Abs   42656 #0A6A0 -  3165  3168  3393
 RFAD97  Abs   42722 #0A6E2 -  3271  3279  3385  3482  3509
 RFAD98  Abs   42734 #0A6EE -  3277  3272
=RFADJ+  Abs   42744 #0A6F8 -  3337
 RFUP++  Abs   42603 #0A66B -  3086  3216
 RFUP-3  Abs   42630 #0A686 -  3101  3098  3104
 RFUP-5  Abs   42632 #0A688 -  3103  3091
=RFUPD+  Abs   42606 #0A66E -  3088  3165  3271  3396  3446  3545  3552  3568
=RNM010  Abs   41583 #0A26F -  1978
 RNM015  Abs   41606 #0A286 -  1987  1984
 RNM050  Abs   41698 #0A2E2 -  2020  2010
 RNM055  Abs   41701 #0A2E5 -  2021  2119
 RNM060  Abs   41705 #0A2E9 -  2022  2035
 RNM070  Abs   41750 #0A316 -  2045  2000  2015  2039  2041  2106  2108  2110
 RNM085  Abs   41709 #0A2ED -  2024  1989
 RNM095  Abs   41764 #0A324 -  2050  2017  2019
=RNMERR  Abs   41754 #0A31A -  2047  2004  2113
 RNMPOL  Abs   41714 #0A2F2 -  2027  1992
 ROMCHK  Ext                -  2444
=ROMF-   Abs   40991 #0A01F -  1271
=ROMF-1  Abs   40988 #0A01C -  1270  1128
 ROMF-2  Abs   41006 #0A02E -  1276  1280
 ROMF-3  Abs   41013 #0A035 -  1278  1275
 ROMFND  Ext                -  2446
 RSTOFS  Abs   41562 #0A25A -  1796  1730  1747
 RSTST   Ext                -   701
 S-CRLF  Ext                -  2539
 S-R0-0  Abs  194673 #2F871 -    12  1472  1488  1783
 S-R0-1  Abs  194678 #2F876 -    12  1701  1751
 S-R1-0  Abs  194689 #2F881 -    12   406   414
 SAVEDO  Ext                -   234
 SAVELO  Ext                -  2743
 SECERR  Abs   41947 #0A3DB -  2362  2350  2391
 SECEXT  Abs   42023 #0A427 -  2390  2352
 SECPOL  Abs   42028 #0A42C -  2394  2354
 SECR02  Abs   41903 #0A3AF -  2346  2341  2442
 SECR10  Abs   41917 #0A3BD -  2351  2347
 SECR15  Abs   41924 #0A3C4 -  2353  2344
 SECR20  Abs   41951 #0A3DF -  2364  2361
 SECR60  Abs   41975 #0A3F7 -  2371  2357
 SECR65  Abs   41978 #0A3FA -  2372  2368
```

```
 SECR70  Abs   42000 #0A410 -  2382  2376
 SECR80  Abs   42015 #0A41F -  2387  2383
 SECRDC  Ext               -  2332  2337
=SECURE  Abs   41889 #0A3A1 -  2340  2335
 SECURP  Ext               -  2333  2338
=SHOW    Abs   42086 #0A466 -  2506
 SHOW10  Abs   42096 #0A470 -  2510  2541
 SHOWp   Ext               -  2504
 TRMPRG  Abs   41062 #0A066 -  1431  1427
 TST12A  Ext               -   666
 UNSEC   Abs   42008 #0A418 -  2385  2379
=UNSECR  Abs   41872 #0A390 -  2334
 UPD1EN  Abs  193945 #2F599 -    12  3163
 UPD1ST  Abs  193885 #2F55D -    12  3162  3163
 UPD2EN  Abs  194214 #2F6A6 -    12  3480
 UPD2ST  Abs  194164 #2F674 -    12  3480  3481
=WRKFIL  Abs   42467 #0A5E3 -  2831  1135  2759
 ZERPGM  Ext               -  1762
=ave=d1  Abs   40772 #09F44 -   902   418
 bserr   Abs   41113 #0A099 -  1450  1425  1440  1454  1486  2043  2362  2822
 csrc5   Ext               -  3367
 eDVCNF  Ext               -  1130  2500
 eFACCS  Ext               -  1670
 eFEXST  Ext               -  2047
 eFSPEC  Ext               -   915
 eFTYPE  Ext               -  2821
 eFnFND  Ext               -  1198
=eolxc+  Abs   41217 #0A101 -  1493  1420  2340  2754
=eolxck  Abs   41220 #0A104 -  1494
 fBASIC  Abs   57876 #0E214 -    11  2792
 fLEX    Abs   57864 #0E208 -    11  1706
 fltdh   Ext               -   692
 flxq18  Abs   40519 #09E47 -   560   539
 flxqpl  Abs   40035 #09C63 -   326   311
 fspcer  Abs   40456 #09E08 -   536   582
 lDATEh  Abs       6 #00006 -    11  2794
 lFIB    Abs      63 #0003F -    11  3555
 lFLAGh  Abs       2 #00002 -    11  2794
 lFLENh  Abs       5 #00005 -    11  2795
 lFNAMh  Abs      16 #00010 -    11  2021  2790  1683  1707  2371  2791
 lFTYPh  Abs       4 #00004 -    11  2793  2372  2794
 lLXADR  Abs       5 #00005 -    11  3569
 lLXID   Abs       2 #00002 -    11  3567
 lLXTKR  Abs       4 #00004 -    11  3567
 lTIMEh  Abs       4 #00004 -    11  2794
 mferr   Abs   41758 #0A31E -  2048  2025  2117  2501
 oBSsod  Abs      17 #00011 -    11  2783
 oDBEGb  Abs      21 #00015 -    11  3533  3546  3553
 oDEVCb  Abs      12 #0000C -    11  3526  3528
 oFBEGb  Abs      13 #0000D -    11  3528  3533  3546
 oFLENh  Abs      32 #00020 -    11  2783
 oFLSTr  Abs      49 #00031 -    11  1732  1772
 oRECLb  Abs      36 #00024 -    11  3553  3554
 oRLENb  Abs      52 #00034 -    11  3554  3555
 pEDIT   Abs      43 #0002B -    11  2817
```

pFILXQ	Abs	3	#00003 -	11	360						
pFPROT	Abs	11	#0000B -	11	2395						
pFSPCx	Abs	5	#00005 -	11	911						
=pPOLL2	Abs	41720	#0A2F8 -	2033	2396						
pPRGPR	Abs	50	#00032 -	11	1666						
pPURGE	Abs	16	#00010 -	11	1453						
pRNAME	Abs	17	#00011 -	11	2028						
poll	Abs	42067	#0A453 -	2498	359	910	1452	1665	2027	2394	2816
=romchk	Abs	42055	#0A447 -	2444	1196	1273	2507	2944			
=romfnd	Abs	42061	#0A44D -	2446	1210	1276	2540	2957			
rstst	Abs	40743	#09F27 -	701	244						
sDEST	Abs	3	#00003 -	11	1979	1997					
tALL	Ext		-	1435							
tCARD	Ext		-	427							
tCOLON	Ext		-	433							
tEOL	Ext		-	2799							
tKEYS	Ext		-	2174							
=tKYSck	Abs	41821	#0A35D -	2173	1433	2346					
tLITRL	Ext		-	437							
tMAIN	Ext		-	431							
tPCRD	Abs	4063727	#E01EF -	11	389						
tPORT	Ext		-	429							
tXWORD	Ext		-	435							

Input Parameters

  Source file name is SG&FXQ::MS

  Listing file name is SG/FXQ:TI:ML::-1

  Object file name is SG%FXQ:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
1            *     SSS   BBBB    &     FFFFF   CCC   N   N
2            *    S   S  B   B  & &    F      C   C  N   N
3            *    S      B   B  & &    F      C      NN  N
4            *     SSS   BBBB    &     FFFF   C      N N N
5            *        S  B   B  & & &  F      C      N  NN
6            *    S   S  B   B  &  &   F      C   C  N   N
7            *     SSS   BBBB    && &  F       CCC   N   N
8            *
9                    TITLE  Function Definitions <831212.1206>
10 0A92B      SBFCN  ABS    #0A92B
11
12
13           ********************************************************************
14           ********************************************************************
15           **
16           ** Name:(S) pVER$   -  VER$ Statement Extension Poll
17           **
18           ** Category:   POLL
19           **
20           ** Type:       FPOLL
21           **
22           ** Purpose:
23           **      Allows a lex file to show its presence and revision
24           **      code.
25           **
26           ** Should poll be "Handled" (return with XM=0)?:
27           **      No!!!
28           **
29           ** Meaning of "Handling" Poll (what does code do if handled?):
30           **      Not applicable
31           **
32           ** Entry conditions for handler (registers, ST, RAM, etc.):
33           **      B[A] = Poll number.
34           **      R2=(AVMEMS)
35           **      R3=Stack pointer
36           **      HEX mode.
37           **      P=0.
38           **
39           ** Normal exit conditions from handler if handled (ST, RAM,
40           ** registers, etc.):
41           **      Not applicable
42           **
43           ** Normal exit conditions from handler if not handled (ST, RAM,
44           ** registers, etc.):
45           **      HEX mode.
46           **      XM=1.
47           **      R2=(AVMEMS)
48           **      R3=New Stack pointer
49           **
50           ** Error exit conditions from handler (POLL only):
51           **      Not applicable
52           **
53           ** Available subroutine levels:
54           **      2
55           **
```

```
56            ** What registers/RAM may be used if handled?:
57            **      Not applicable.
58            **
59            ** What registers/RAM may be used if not handled?:
60            **      A-C, D[15-5] D0, D1, P
61            **      NOTE: D[A] is sacred in FPOLL!!
62            **      R1 and R4.  Function scratch is available in the
63            **      unlikely event it it needed.
64            **
65            ** What registers/RAM may be used if error exit (POLL only)?:
66            **      Not applicable
67            **
68            ** Special memory/pointer considerations (are pointers funny?):
69            **      This occurs during expression execute so keep in mind
70            **      the rules of that game.
71            **
72            ** Envisioned application(s):
73            **      The poll handler is expected to add onto the string
74            **      being built on the stack.  The stack pointer is kept
75            **      in R3 and must be decremented to point to the new
76            **      end of the string.  Available memory should be checked
77            **      by comparing against the AVMEMS (which resides in R2).
78            **
79            **      The string added should have a leading blank followed
80            **      by a short (~3-5 characters) name describing the lex
81            **      file and optionally followed by a colon and a revision
82            **      code.  The revision code will usually be just a digit
83            **      but a more complicated code may be required for a
84            **      multi-chip ROM.
85            **
86            ** History:
87            **
88            **    Date     Programmer            Modification
89            **  --------   ----------   ----------------------------------
90            **  06/08/83   B.S.         Added documentation.
91            **
92            ********************************************************************
93            ********************************************************************
94            ********************************************************************
95            ********************************************************************
96            **
97            ** Name:    VER$    -  VER$ function
98            **
99            ** Category:   FNEXEC
100           **
101           ** Purpose:
102           **      Implements the VER$ function
103           **
104           ** Entry:
105           **      P      =  0
106           **      D1 points to top of stack
107           **      D0 is program pointer (points past VER$ token)
108           **
109           ** Exit:
110           **      P      =  0
```

```
111             **      Exits thru EXPR
112             **
113             ** Calls:      XXHEAD,BF2STK,STKCHR,FPOLL,D=AVMS,ADHEAD,EXPR
114             **
115             ** Stk lvls:   2
116             **
117             ** History:
118             **
119             **    Date     Programmer             Modification
120             **  --------   ----------   ----------------------------------
121             **  07/28/83   B.S.         Added documentation
122             **
123             ********************************************************************
124             ********************************************************************
125 0A92B         VER"
126 0A92B 8405          NIBASC \HP71:1\
          7313
          A313
127 0A937 FF            NIBHEX FF
128
129 0A939 00            NIBHEX 00
130 0A93B 34B2 =VER$    LC(5)  VER"
          9A0
131 0A942 850           ST=1   0                Return from BF2STK
132 0A945 8F00          GOSBVL =BF2STK          Put buffer onto stack
          000
133 0A94C 8F00          GOSBVL =XXHEAD          Remove stack header
          000
134 0A953 136           CDOEX
135 0A956 108           R0=C                    Save PC
136
137 0A959 1B00          D0=(5) =R1REV
          000
138 0A960 7D40          GOSUB  REVCHR           Get ROM 1 rev
139 0A964 1A00          D0=(4) =R2REV
          00
140 0A96A 7340          GOSUB  REVCHR           Get ROM 2 rev
141 0A96E 1B00          D0=(5) =R3REV
          000
142 0A975 7830          GOSUB  REVCHR           Get ROM 3 rev
143 0A979 1A00          D0=(4) =R4REV
          00
144 0A97F 7E20          GOSUB  REVCHR           Get ROM 4 rev
145
146 0A983 DB            C=D    A
147 0A985 10A           R2=C                    R2=(AVMEMS) for ROMs to use
148 0A988 137           CD1EX
149 0A98B 10B           R3=C                    Save Stack pointer
150 0A98E 8E00          GOSUBL =FPOLL           Poll ROMs for their versions
          00
151 0A994 00            CON(2) =pVER$
152 0A996 118           C=R0
153 0A999 134           D0=C                    Restore PC
154 0A99C 8F00          GOSBVL =D=AVMS
          000
```

```
155 0A9A3 11B          C=R3
156 0A9A6 135          D1=C                    Restore stack pointer
157 0A9A9 7690         GOSUB  Adhead
158 0A9AD 6C30         GOTO   EXPRj
159             *-
160             *-
161 0A9B1 14E   REVCHR C=DAT0 B
162 0A9B4 8D00 =stkchr GOVLNG =STKCHR
          000
163             *-
164             *-
```

```
165                     STITLE NUM
166                *****************************************************************
167                *****************************************************************
168                **
169                ** Name:    NUM     -  NUM function execution
170                **
171                ** Category:   FNEXEC
172                **
173                ** Purpose:
174                **      Implements the NUM function
175                **
176                ** Entry:
177                **      P       =  0
178                **      D1      =  Stack pointer
179                **      D0      =  Program counter (points past NUM token)
180                **
181                ** Exit:
182                **      D1 updated to new top of stack
183                **      Exits thru EXPR
184                **
185                ** Calls:      POP1S,HDFLT,EXPR
186                **
187                ** Stk lvls:   1
188                **
189                ** History:
190                **
191                **    Date      Programmer              Modification
192                **  --------   ----------   ------------------------------------
193                **  07/28/83   B.S.         Added documentation
194                **
195                *****************************************************************
196                *****************************************************************
197 0A9BB 411              NIBHEX 411
198 0A9BE 8E00 =NUM       GOSUBL =POP1S          Check for string on stack
           00
199 0A9C4 137              CD1EX
200 0A9C7 C2               C=C+A  A
201 0A9C9 135              D1=C                   Pop string off stack
202 0A9CC AF8              B=A    W               B(A)=string length
203 0A9CF AF0              A=0    W               Zero
204 0A9D2 8A9              ?B=0   A               Null string?
205 0A9D5 B1               GOYES  NUM20           Yes, then return 0
206 0A9D7 1C1              D1=D1- 2               Point to first character
207 0A9DA 14B              A=DAT1 B               Read in first char
208 0A9DD 1CD              D1=D1- 14              Set ptr to write out numeric item
209 0A9E0 8E00             GOSUBL =hdflt          Convert byte to decimal float
           00
210 0A9E6 1517  NUM10      DAT1=A W               Write out NUM value
211 0A9EA 8C00 =EXPRJ      GOLONG =EXPR
           00
212                *-
213                *-
214 0A9F0 1CF   NUM20      D1=D1- 16              Set ptr to write out num item(0)
215 0A9F3 52F              GONC   NUM10           (B.E.T.)
216                *-
```

```
217             *-
```

```
218                     STITLE CHARSET$
219                 ********************************************************************
220                 ********************************************************************
221                 **
222                 ** Name:    CHRST$  -  CHARSET$ function
223                 **
224                 ** Category:   FNEXEC
225                 **
226                 ** Purpose:
227                 **      Implements the CHARSET$ function
228                 **
229                 ** Entry:
230                 **      P       =  0
231                 **      D1 is stack pointer
232                 **      D0 is program counter (points past CHARSET$ token)
233                 **
234                 ** Exit:
235                 **      P       =  0
236                 **      D1 is new stack pointer
237                 **      Exits through EXPR
238                 **
239                 ** Calls:      D=AVMS,IOFNDO,CHRBUF,ADHEAD
240                 **
241                 ** Stk lvls:   1
242                 **
243                 ** History:
244                 **
245                 **    Date      Programmer              Modification
246                 **  --------   ----------   ----------------------------------
247                 **  07/28/83   B.S.         Added documentation
248                 **
249                 ********************************************************************
250                 ********************************************************************
251 0A9F6 00              NIBHEX 00
252 0A9F8 8F00  =CHRST$ GOSBVL =D=AVMS
            000
253 0A9FF D5            B=C    A               Save stack pointer
254 0AA01 109           R1=C                   Save it for ADHEAD too.
255 0AA04 3200          LC(3)  =bALTCH
            0
256 0AA09 8E00          GOSUBL =IOFNDO         Find buffer
            00
257 0AA0F 440           GOC    CHRS$2          Found?
258 0AA12 D0            A=0    A               No, then return null string
259 0AA14 8F00  CHRS$2 GOSBVL =CHRBUF         Find actual character set
            000
260 0AA1B 137           CD1EX                  C(A)=Buffer ptr, D1=stack ptr
261 0AA1E 136           CD0EX                  C(A)=PC,D0=Buffer pointer
262 0AA21 108           R0=C                   R0 saves PC
263 0AA24 81C           ASRB                   Divide length by 2 for bytes
264 0AA27 A3C   CHRS$3 A=A-1  X               Decrement byte count
265 0AA2A 4F0           GOC    CHRS$4
266 0AA2D 14E           C=DAT0 B               Read byte from buffer
267 0AA30 161           D0=D0+ 2
268 0AA33 7D7F          GOSUB  stkchr          Add byte to stack
```

```
269 0AA37 5FE          GONC   CHRS$3         (B.E.T.)
270 0AA3A 118   CHRS$4 C=R0                   Recall PC
271 0AA3D 134          D0=C                  Restore PC to D0
272 0AA40 840          ST=0   0              Don't return from ADHEAD
273 0AA43 8D00 =Adhead GOVLNG =ADHEAD
          000
```

```
                        STITLE UPRC$
                *****************************************************************
                *****************************************************************
                **
                ** Name:    UPRC$   -  UPRC$ function
                **
                ** Category:   FNEXEC
                **
                ** Purpose:
                **      Implements the UPRC$ function
                **
                ** Entry:
                **      P      =  0
                **      D1 is stack pointer
                **      D0 is program counter (Points past UPRC$ token)
                **
                ** Exit:
                **      P      =  0
                **      D1 is new stack pointer
                **      Exits thru EXPR
                **
                ** Calls:      POP1S,CNVUCR,EXPR
                **
                ** Stk lvls:   1
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------   ----------   -----------------------------------
                **  07/28/83   B.S.         Added documentation
                **
                *****************************************************************
                *****************************************************************
                *                                  Upper case conversion
OAA4A 411               NIBHEX 411
OAA4D 137   =UPRC$      CD1EX
OAA50 135               D1=C
OAA53 D7                D=C    A
OAA55 8E00              GOSUBL =POP1S
      00
OAA5B AF1               B=0    W
OAA5E D8                B=A    A
OAA60 81D               BSRB                    Convert to bytes
OAA63 CD     UPRC10     B=B-1  A                Decrement loop counter
OAA65 421               GOC    UPRC20
OAA68 8F00              GOSBVL =CNVUCR          A=DAT1 B,Convert to upper case
      000
OAA6F 149               DAT1=A B
OAA72 171               D1=D1+ 2
OAA75 5DE               GONC   UPRC10           (B.E.T.)
                *-
                *-
OAA78 DB     UPRC20     C=D    A
OAA7A 135               D1=C
OAA7D 6C6F              GOTO   EXPRJ
```

```
    327 0AA81              END
```

```
 ADHEAD  Ext                    -  273
=Adhead  Abs   43587 #0AA43    -  273   157
 BF2STK  Ext                    -  132
 CHRBUF  Ext                    -  259
 CHRS$2  Abs   43540 #0AA14    -  259   257
 CHRS$3  Abs   43559 #0AA27    -  264   269
 CHRS$4  Abs   43578 #0AA3A    -  270   265
=CHRST$  Abs   43512 #0A9F8    -  252
 CNVUCR  Ext                    -  318
 D=AVMS  Ext                    -  154   252
 EXPR    Ext                    -  211
=EXPRj   Abs   43498 #0A9EA    -  211   158   326
 FPOLL   Ext                    -  150
 IOFNDO  Ext                    -  256
=NUM     Abs   43454 #0A9BE    -  198
 NUM10   Abs   43494 #0A9E6    -  210   215
 NUM20   Abs   43504 #0A9F0    -  214   205
 POP1S   Ext                    -  198   312
 R1REV   Ext                    -  137
 R2REV   Ext                    -  139
 R3REV   Ext                    -  141
 R4REV   Ext                    -  143
 REVCHR  Abs   43441 #0A9B1    -  161   138   140   142   144
 STKCHR  Ext                    -  162
=UPRC$   Abs   43597 #0AA4D    -  309
 UPRC10  Abs   43619 #0AA63    -  316   321
 UPRC20  Abs   43640 #0AA78    -  324   317
 VER"    Abs   43307 #0A92B    -  125   130
=VER$    Abs   43323 #0A93B    -  130
 XXHEAD  Ext                    -  133
 bALTCH  Ext                    -  255
 hdflt   Ext                    -  209
 pVER$   Ext                    -  151
=stkchr  Abs   43444 #0A9B4    -  162   268
```

Input Parameters

  Source file name is SB&FCN::MS

  Listing file name is SB/FCN:TI:ML::-1

  Object file name is SBXFCN:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
      1                    TITLE  Zero File - End of chain
      2 00000 00           NIBHEX 00
      3 00002              END
```

Input Parameters

  Source file name is JP&ZER::MS

  Listing file name is JP/ZER:TI:ML::-1

  Object file name is JP%ZER:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    TTTTT  III   &     RRRR   EEEEE  V   V
 2            *      T     I   & &    R   R  E      V   V
 3            *      T     I   & &    R   R  E      V   V
 4            *      T     I    &     RRRR   EEEE    V V
 5            *      T     I   & & &  R R    E       V V
 6            *      T     I   &  &   R  R   E        V
 7            *      T    III   && &  R   R  EEEEE    V
 8
 9                     TITLE  HP-71 Revision Number
10 00000 24           NIBASC \B\
11 00002              END
```

Input Parameters

  Source file name is TI&REV::MS

  Listing file name is TI/REV:TI:ML::-1

  Object file name is TI%REV:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    TTTTT  III   &     FFFFF  X   X   222
 2            *      T     I   & &    F      X   X  2   2
 3            *      T     I   & &    F       X X       2
 4            *      T     I    &     FFFF     X      22
 5            *      T     I   & & &  F       X X    2
 6            *      T     I   &  &   F      X   X  2
 7            *      T    III   && &  F      X   X  22222
 8            *
 9                     TITLE ROM 2 Fix Module
10 0AA85             ABS    #0AA85
11            *      TITLE  BUGFIX MODULE
12
13 0AA85 0B0 =B9605A NIBHEX 0B0           NUMERIC, BASE 0, DIM 1
14 0AA88 097         NIBHEX 097           NUMERIC, BASE 0, DIM 2
15 0AA8B 0A0         NIBHEX 0A0           NUMERIC, BASE 1, DIM 1
16 0AA8E 046         NIBHEX 046           NUMERIC, BASE 1, DIM 2
17 0AA91 CE2         NIBHEX CE2           STRING, BASE 0, DIM 1
18 0AA94 440         NIBHEX 440           STRING, BASE 0, SIMPLE
19 0AA97 8A2         NIBHEX 8A2           STRING, BASE 1, DIM 1
20 0AA9A 440         NIBHEX 440           STRING, BASE 1, SIMPLE
21
22 0AA9D D6  =B9605B C=A    A
23 0AA9F 8F00        GOSBVL =CSRC4
         000
24 0AAA6 20          P=     0
25 0AAA8 36F8        LCHEX  000008F
         0000
         0
26 0AAB1 02          RTNSC
27            *
28            * Fix to SR# 1040-5
29            *   SST a RETURN with ELSE following the GOSUB
30            *
31            *   Update PCADDR @ RETURN address
32            *   Position PCADDR @ the line length
33            *     Requires skipping over EOL or "@"
34            *
35 0AAB3 134 =UPDPCA D0=C                 D0 @ Return address
36 0AAB6 14A         A=DAT0 B             Read EOL or @
37 0AAB9 161         D0=D0+ 2             Move past EOL or @
38 0AABC 20          P=     0
39 0AABE 90C         ?A#0   P             Not EOL ?
40 0AAC1 50          GOYES  UPDPC3
41 0AAC3 163         D0=D0+ 4             Skip over Line number
42 0AAC6 136  UPDPC3 CD0EX                Move address to C
43 0AAC9 8C00        GOLONG =UPDPCC       Update PCADDR
         00
```

```
=B9605A  Abs   43653 #0AA85 -    13
=B9605B  Abs   43677 #0AA9D -    22
 CSRC4   Ext                -    23
 UPDPC3  Abs   43718 #0AAC6 -    42    40
=UPDPCA  Abs   43699 #0AAB3 -    35
 UPDPCC  Ext                -    43
```

Input Parameters

  Source file name is TI&FX2::MS

  Listing file name is TI/FX2:TI:ML::-1

  Object file name is TI%FX2:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *      A    BBBB    &     RRRR   EEEEE   GGG
 2            *     A A   B   B  & &    R   R  E      G   G
 3            *    A   A  B   B  & &    R   R  E      G
 4            *    A   A  BBBB    &     RRRR   EEEE   G GGG
 5            *    AAAAA  B   B  & & &  R R    E      G   G
 6            *    A   A  B   B  &  &   R  R   E      G   G
 7            *    A   A  BBBB    && &  R   R  EEEEE   GGG
 8                      TITLE  Variable Creation Routines <831216.1818>
 9 0ABF8                ABS    #0ABF8
10            *
11                      RDSYMB TIXEQU::MS
12            *
13             Array  EQU    0
14             InDEF  EQU    0
15             String EQU    1
16             NonEx  EQU    2
17             OpBase EQU    3
18             NoZero EQU    3
19            ********************************************************************
20            ********************************************************************
21            **
22            ** Name:     INTEGR  -  Dimension Variables
23            ** Name:     SHORT   -  Dimension Variables
24            ** Name:     REAL    -  Dimension Variables
25            ** Name:     DIM     -  Dimension Variables
26            **
27            ** Category:   STEXEC
28            **
29            ** Purpose:
30            **      Execute INTEGER, SHORT, REAL and DIM statements.
31            **
32            ** Entry:
33            **      DO=PC, pointing past INTEGER, SHORT, REAL or DIM token.
34            **      Jumped on said token.
35            **      P=0.
36            **
37            ** Exit:
38            **      Through NXTSTM.
39            **
40            ** Calls:      PREP, DPVCTR, SPACE, DMNSN.
41            **
42            ** Detail:
43            **   SYNTAX:  DIM A,A1(15),A2(3,6),B$,B1$[26],B2$(5)[30]
44            **
45            **   THE FOLLOWING TOKENS ARE EXPECTED:
46            **      Simple Variable - tDIM tVAR tCOMMA
47            **      Numeric Array   - tDIM tARRAY tVAR EXPRESSION tCOMMA
48            **      String          - tDIM tVAR ; EXPRESSION tCOMMA
49            **      String Vector   - tDIM tARRAY tVAR EXPRESSION ;
50            **                          EXPRESSION tCOMMA
51            **
52            ** History:
53            **
54            **    Date     Programmer              Modification
55            **  --------   ----------   ----------------------------------
```

```
                **              SA          Wrote
                **  10/18/83    NM          Attempted to document
                **
                ********************************************************************
                ********************************************************************
0ABF8 0000              REL(5) =DECDC
      0
0ABFD 0000              REL(5) =DECP            PARSE ROUTINE FOR REAL
      0
0AC02             =INTEGR
0AC02             =SHORT
0AC02 6D00        =REAL   GOTO   DIMSTM         ENTRY POINT FOR REAL

0AC06 0000              REL(5) =DECDC
      0
0AC0B 0000              REL(5) =DIMP
      0
0AC10             =DIM
0AC10 181         =DIMSTM D0=D0- 2
0AC13 14E               C=DAT0 B              REREAD TOKEN
0AC16 161               D0=D0+ 2
0AC19 1F00              D1=(5) =S-R1-3
      000
0AC20 1550              DAT1=C P              STORE DATA TYPE INDICATOR
0AC24 7781  DMSTM1      GOSUB  PREP           INITIALIZE
0AC28 7420              GOSUB  DPVCTR         CREATE DOPE VECTOR
0AC2C 109               R1=C
0AC2F 7A61              GOSUB  SPACE          COMPUTE SPACE REQUIREMENTS
0AC33 7202              GOSUB  DMNSN          DIMENSION VARIABLE
0AC37 4CE               GOC    DMSTM1         IF NOT AT END, REPEAT DMSTM1
0AC3A 6424              GOTO   CRTRTN         REPEAT NXTSTM
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) LIMITS  -  Compute Dimension Limits In Decl Stmt
                **
                ** Category:   EXCUTL
                **
                ** Purpose:
                **      Compute the dimension limits in a declaration statement
                **      (INTEGER, REAL, SHORT, DIM).  Collapses the stack
                **      beforehand.
                **
                ** Entry:
                **      D0 pointing at start of tokenized expression.
                **
                ** Exit:
                **      D0 pointing past expression.
                **      D1 @ top of math stack.
                **
                ** Calls:      COLLAP, EXPEX+.
                **
                ** Uses.......
                **              Everything available to expression execute.
                **
```

```
106                ** Stk lvls:   5
107                **
108                ** History:
109                **
110                **    Date     Programmer            Modification
111                **  --------   ----------   ----------------------------------
112                **             SA           Wrote
113                **  10/18/83   NM           Attempted to document
114                **
115                ********************************************************************
116                ********************************************************************
117 0AC3E 8E00 =LIMITS GOSUBL =COLLAP
           00
118 0AC44 8E00         GOSUBL =EXPEX+
           00
119 0AC4A 8C00         GOLONG =RSTST
           00
120                ********************************************************************
121                ********************************************************************
122                **
123                ** Name:(S) DPVCTR  -  Creates Vars, Computes # Of Elements
124                **
125                ** Category:   VARMGT
126                **
127                ** Purpose:
128                **      Creates primary variables(dope vectors), computes
129                **      number of array units to allocate
130                **
131                ** Entry:
132                **      Same as exit conditions from PREP, ie
133                **      P      = 0
134                **      D0 points to dimension expression(s) if array
135                **      A(X),(S-R1-2) = 3-digit code for variable
136                **      B(A),(S-R0-0) = Address of variable(if it exists(S2=0))
137                **      (S-R0-1 thru S-R1-1) zeroed
138                **      Array(S0) set iff it is an array
139                **      NonEx(S2) set iff variable/array doesn't already exist
140                **      String(S1) set iff string variable/array
141                **      OpBase(S3) set iff OPTION BASE 1
142                **
143                ** Exit:
144                **      P      = 0
145                **      C-register has the following information:
146                **
147                **         +------------+----------+----------+--+--+--+
148                **         | ////////// |dimlimt 1 |dimlimt 2 | b| d| t|
149                **         +------------+----------+----------+--+--+--+
150                **                           4          4        1  1  1
151                **      where t is datatype indicator
152                **            d is dimcount
153                **            b is baseoption
154                **            dimlimt 2 is second dimlimit or max string length
155                **            dimlimt 1 is first dimlimit
156                **      or
157                **
```

```
158                **        +---------------------------------------+--+
159                **        |              zeroes                   | t|
160                **        +---------------------------------------+--+
161                **
162                **      where t is datatype indicator (0 for real)
163                **      for real, short, and integer simple variables.
164                **
165                **         A(A) = number of array units
166                **         B(X) = 3-nibble code for variable
167                **         S-R0-1 = 1st subscript if is an array
168                **         S-R0-2 = 2nd subscript if is a 2 dimensional array
169                **                = Maximum string length if string
170                **         S-R1-0 = Number of elements for numeric array
171                **
172                ** Calls:      LIMITS,GETDIM,A-MULT
173                **
174                ** Uses.......
175                **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D0,D1
176                **
177                ** Stk lvls:   6
178                **
179                ** History:
180                **
181                **    Date      Programmer              Modification
182                **  --------   ----------   ----------------------------------
183                **             SA           Wrote
184                **
185                ****************************************************************
186                ****************************************************************
187 0AC50 AC3  =DPVCTR D=0    S
188 0AC53 860         ?ST=0   Array          SIMPLE VARIABLE?
189 0AC56 53          GOYES   VCT10          IF SO, GOTO VCT10
190 0AC58 72EF        GOSUB   LIMITS         COMPUTE DIMLIMITS
191 0AC5C AC3         D=0     S
192 0AC5F B47         D=D+1   S              INCREMENT DIMCOUNT
193 0AC62 7501        GOSUB   GETDIM         GET DIMLIMIT
194 0AC66 17F         D1=D1+  16
195 0AC69 D7          D=C     A
196 0AC6B 137         CD1EX
197 0AC6E 1F00        D1=(5)  =FORSTK
            000
198 0AC75 143         A=DAT1  A
199 0AC78 8A2         ?A=C    A              STACK EXHAUSTED?
200 0AC7B 01          GOYES   VCT10          IF NOT, GOTO VCT10
201 0AC7D 137         CD1EX
202 0AC80 B47         D=D+1   S              INCREMENT DIMCOUNT
203 0AC83 74E0        GOSUB   GETDIM         GET DIMLIMIT
204 0AC87 DF          CDEX    A              SWAP WITH PREVIOUS DIMLIMIT
205 0AC89 DA          A=C     A
206 0AC8B 1F00  VCT10 D1=(5)  =S-R0-3
            000
207 0AC92 AFB         C=D     W
208 0AC95 1554        DAT1=C  S              SAVE DIMCOUNT IN S-R0-3
209 0AC99 1C4         D1=D1-  5
210 0AC9C 145         DAT1=C  A              SAVE FIRST DIMLIMIT IN S-R0-2
```

```
211 0AC9F 1C4          D1=D1- 5
212 0ACA2 141          DAT1=A A              SAVE SECOND DIMLIMIT IN S-R0-1
213 0ACA5 861          ?ST=0  String         STRING?
214 0ACA8 82           GOYES  VCT20          IF NOT, GOTO VCT20
215 0ACAA 14A          A=DATO B              READ DELIMITER
216 0ACAD F0           ASL    A
217 0ACAF 3402         LCHEX  00020          LOAD DEFAULT STRING LENGTH
          000
218 0ACB6 966          ?A#C   B              SEMICOLON TOKEN?
219 0ACB9 D0           GOYES  VCT15          IF NOT, GOTO VCT15
220 0ACBB 161          D0=D0+ 2
221 0ACBE 7C7F         GOSUB  LIMITS         COMPUTE STRING LENGTH
222 0ACC2 75A0         GOSUB  GETDIM
223 0ACC6 1F00 VCT15   D1=(5) =S-R0-1
          000
224 0ACCD 145          DAT1=C A              STORE STRING LENGTH
225 0ACD0 147  VCT20   C=DAT1 A              READ SECOND DIMLIMIT
226 0ACD3 174          D1=D1+ 5
227 0ACD6 143          A=DAT1 A              READ FIRST DIMLIMIT
228 0ACD9 174          D1=D1+ 5
229 0ACDC 1534         A=DAT1 S              READ DIMCOUNT
230 0ACE0 17F          D1=D1+ 16
231 0ACE3 1574         C=DAT1 S              READ DATATYPE
232 0ACE7 873          ?ST=1  OpBase         BASE OPTION ZERO?
233 0ACEA B0           GOYES  VCT30          IF NOT, GOTO VCT30
234 0ACEC E4           A=A+1  A              INCREMENT DIMLIMIT
235 0ACEE 871          ?ST=1  String         STRING?
236 0ACF1 40           GOYES  VCT30          IF SO, GOTO VCT30
237 0ACF3 E6           C=C+1  A              INCREMENT DIMLIMIT
238 0ACF5 861  VCT30   ?ST=0  String         STRING?
239 0ACF8 41           GOYES  VCT40          IF NOT, GOTO VCT40
240 0ACFA 850          ST=1   Array
241 0ACFD DE           ACEX   A
242 0ACFF E4           A=A+1  A
243 0AD01 E4           A=A+1  A
244 0AD03 B44          A=A+1  S
245 0AD06 AC2          C=0    S
246 0AD09 A4E          C=C-1  S
247 0AD0C A4C  VCT40   A=A-1  S              SIMPLE NUMERIC VARIABLE?
248 0AD0F 4B3          GOC    VCT60          IF SO, GOTO VCT60
249 0AD12 1CE          D1=D1- 15
250 0AD15 948          ?A=0   S              ONE DIMENSION?
251 0AD18 B0           GOYES  VCT50          IF SO, GOTO VCT50
252 0AD1A 8E00         GOSUBL =a-mult        COMPUTE NUMBER OF ELEMENTS
          00
253 0AD20 5E3          GONC   TOOBIG         IF TOO BIG, GOTO TOOBIG
254 0AD23 141  VCT50   DAT1=A A              SAVE NUMBER OF ELEMENTS IN S-R1-0
255 0AD26 1CB          D1=D1- 12
256 0AD29 15F9         C=DAT1 10             READ 1ST DIMLIMIT TO A(9-6)
257 0AD2D 1C0          D1=D1- 1
258 0AD30 15F5         C=DAT1 6              READ 2ND DIMLIMIT TO A(5-2)
259 0AD34 17B          D1=D1+ 12
260 0AD37 1570         C=DAT1 P              READ DIMCOUNT
261 0AD3B 812          CSLC
262 0AD3E AA2          C=0    XS
```

```
 263 0AD41 863          ?ST=0  OpBase         BASEOPTION ZERO?
 264 0AD44 00           RTNYES                IF SO, RETURN CARRY SET
 265 0AD46 B26          C=C+1  XS
 266 0AD49 02           RTNSC
 267 0AD4B D0    VCT60  A=0    A
 268 0AD4D 80DF  VCT70  P=C    15             YES, BUT IT'S SHORTER
 269 0AD51 AF2          C=0    W
 270 0AD54 89C          ?P=    12
 271 0AD57 6F           GOYES  VCT70
 272 0AD59 80F0         CPEX   0
 273 0AD5D 02           RTNSC
 274            ****************************************************************
 275 0AD5F 8C00  TOOBIG GOLONG =MEMERR
          00
 276 0AD65 8C00  DATATP GOLONG =CNFLCT
          00
 277            ****************************************************************
 278            ****************************************************************
 279            **
 280            ** Name:(S) GETDIM  -  Get A Dimlimit From Stack
 281            **
 282            ** Category:   VARMGT
 283            **
 284            ** Purpose:
 285            **      Pop dimension limit from stack and check range.
 286            **
 287            ** Entry:
 288            **      D1=stack pointer.
 289            **
 290            ** Exit:
 291            **      P=0.
 292            **      HEX mode.
 293            **      Errors out if result complex (eDATTY) or out of range
 294            **        (eARGOR).
 295            **      A[A]=dimlimit.
 296            **
 297            ** Calls:      FLTDH, POP1N.
 298            **
 299            ** Uses.......
 300            **             A,B,C,P.
 301            **
 302            ** Stk lvls:   2
 303            **
 304            ** History:
 305            **
 306            **    Date     Programmer              Modification
 307            **  --------   ----------   ----------------------------------
 308            **             SA           Wrote
 309            **  10/18/83   NM           Attempted to document
 310            **
 311            ****************************************************************
 312            ****************************************************************
 313 0AD6B 8E00 =GETDIM GOSUBL =POP1N         POP DIMLIMIT
          00
 314 0AD71 04           SETHEX                COMPLEX NUMBER?
```

```
0AD73 41F          GOC    DATATP          IF SO, GOTO DATATP
0AD76 8E00         GOSUBL =fltdh          CONVERT TO HEX INTEGER
      00
0AD7C 470          GOC    GTDM10          IF OK, GOTO GTDM10
0AD7F 8AC          ?A#0   A               DIMLIMIT = 0?
0AD82 71           GOYES  ARGORL          IF NOT, GOTO ARGORL
0AD84 863  GTDM10  ?ST=0  OpBase          BASEOPTION ZERO?
0AD87 70           GOYES  GTDM20          IF SO, GOTO GTDM20
0AD89 8A8          ?A=0   A               DIMLIMIT = 0?
0AD8C D0           GOYES  ARGORL          IF SO, GOTO ARGORL
0AD8E D6   GTDM20  C=A    A
0AD90 F2           CSL    A
0AD92 F6           CSR    A
0AD94 8A2          ?A=C   A               DIMLIMIT < 2^16?
0AD97 00           RTNYES                 IF SO, RETURN CARRY SET
0AD99 6BC2 ARGORL GOTO    ARGOR          SHORT JUMP LINKAGE
           ********************************************************************
           ********************************************************************
           **
           ** Name:(S) SPACE   -  Compute Space Needs For An Array
           **
           ** Category:   VARMGT
           **
           ** Purpose:
           **      Calculate space requirements for an array.
           **
           ** Entry:
           **      P=0.
           **      A[A] = number of array units needed.
           **      C[0] = data type:
           **             A - Integer
           **             B - Short real
           **             C - Real
           **             D - Short complex
           **             E - Complex
           **      Error exit (eMEM) if > address space.
           **
           ** Exit:
           **      A,R0 = space requirements in nibbles.
           **      P=0.
           **
           ** Calls:      LENGTH, A-MULT.
           **
           ** Uses.......
           **             A,B,C,R0.
           **
           ** Stk lvls:   1
           **
           ** History:
           **
           **    Date     Programmer            Modification
           **  --------   ----------   ----------------------------------
           **             SA           Wrote
           **  10/18/83   NM           Attempted to document
           **
```

```
369           ****************************************************************
370           ****************************************************************
371 0AD9D 70E7 =SPACE  GOSUB  LENGTH        COMPUTE SPC REQTS IN NIBS
372 0ADA1 8E00        GOSUBL =a-mult
          00
373 0ADA7 57B         GONC   TOOBIG
374 0ADAA 100         R0=A
375 0ADAD 01          RTN
376           ****************************************************************
377           ****************************************************************
378           **
379           ** Name:(S) PREP    -  Prepare To Create A Variable/array
380           **
381           ** Category:   VARMGT
382           **
383           ** Purpose:
384           **      Prepare to create a variable or array
385           **
386           ** Entry:
387           **      D0 points to tokenization of a variable or array in
388           **      some "dim" statement.
389           **
390           ** Exit:
391           **      P      =  0
392           **      A(X),(S-R1-2) = 3-digit code for variable
393           **      B(A),(S-R0-0) = Address of variable(if it exists(S2=0))
394           **      (S-R0-1 thru S-R1-1) zeroed
395           **      Array(S0) set iff it is an array
396           **      NonEx(S2) set iff variable/array doesn't already exist
397           **      String(S1) set iff string variable/array
398           **      Carry and OpBase(S3) set iff OPTION BASE 1
399           **
400           ** Calls:      ADDRSS,C=ACTV,BASE
401           **
402           ** Uses.......
403           **  Inclusive: D0,D1,S0,S1,S2,S3,A(A),B(A),C(W),D(A)
404           **
405           ** Stk lvls:   2
406           **
407           ** Note:  Takes error exit if trying to change a function
408           **        parameter.
409           **
410           ** History:
411           **
412           **    Date     Programmer              Modification
413           **  --------   ----------   ----------------------------------
414           **             SA           Wrote
415           **
416           ****************************************************************
417           ****************************************************************
418 0ADAF 04   =PREP   SETHEX
419 0ADB1 20          P=     0
420 0ADB3 840         ST=0   Array
421 0ADB6 14A         A=DAT0 B
422 0ADB9 3100        LC(2)  =tARRAY
```

```
423 OADBD 966          ?A#C   B              ARRAY DECLARATION ?
424 OADC0 80           GOYES  PREP10         IF NOT GOTO PREP10
425 OADC2 850          ST=1   Array          SET ARRAY FLAG
426 OADC5 161          DO=DO+ 2              PASS THE ARRAY TOKEN
427 OADC8 841  PREP10  ST=0   String
428 OADCB 14A          A=DAT0 B
429 OADCE 31D2         LCHEX  2D
430 OADD2 966          ?A#C   B              STRING DECLARATION?
431 OADD5 50           GOYES  PREP20         IF NOT, GOTO PREP20
432 OADD7 851          ST=1   String         SET STRING FLAG
433 OADDA 852  PREP20  ST=1   NonEx
434 OADDD 8E00         GOSUBL =ADDRSS        SEARCH FOR THE VARIABLE
          00
435 OADE3 132          ADOEX
436 OADE6 401          GOC    PREP30         IF NOT FOUND, GOTO PREP30
437 OADE9 842          ST=0   NonEx          DENOTE EXISTING VARIABLE
438 OADEC D5           B=C    A              SAVE NAME IN B
439 OADEE 7EA4         GOSUB  C=ACTV
440 OADF2 8B2          ?A<C   A              FUNCTION PARAMETER?
441 OADF5 83           GOYES  NORDIM         IF SO, GOTO NORDIM
442 OADF7 1F00 PREP30  D1=(5) =S-R1-2
          000
443 OADFE DC           ABEX   A              LEAVE ADDRESS IN B
444 OAE00 141          DAT1=A A              SAVE NAME IN S-R1-2
445 OAE03 AF2          C=0    W
446 OAE06 1C9          D1=D1- 10
447 OAE09 15D9         DAT1=C 10             CLEAR STUFF
448 OAE0D 1CF          D1=D1- 16
449 OAE10 D9           C=B    A
450 OAE12 1557         DAT1=C W              SAVE ADDRESS IN S-R0-0
451 OAE16 132          ADOEX
452 OAE19 843          ST=0   OpBase
453 OAE1C 8E00         GOSUBL =BASE
          00
454 OAE22 550          GONC   PREP40         SET BASEOPTION FLAG
455 OAE25 853          ST=1   OpBase
456 OAE28 132  PREP40  ADOEX
457 OAE2B 01           RTN                   RETURN
458             *
459             ********************************************************************
460             ********************************************************************
461             **
462             ** Name:(S) NORDIM  -  Report "Var Context" Error
463             **
464             ** Category:   SYSTEM
465             **
466             ** Purpose:
467             **      Report "Var Context" as an execution error.
468             **
469             ** Entry:
470             **      P      =  0
471             **      S13=0 if not a running program (i.e., keyboard
472             **         execution error)
473             **      S13=1 if running program
474             **
```

```
475             ** Exit:
476             **      Exits to BASIC main loopo (ERRRTN)
477             **
478             ** Calls:       MFERR
479             **
480             ** Uses.......  BASIC main loop can use anything
481             **
482             ** Stk lvls:    BASIC main loop can use all
483             **
484             ** NOTE:
485             **      Setting P=0 selects the following error options:
486             **        -- not a parse error
487             **        -- store ERRN (and ERRL if S13=1)
488             **        -- display "ERR:" (or "ERR L<#>:")
489             **
490             ** Detail:
491             **     =NORDIM LC(2)  =eVCNTX
492             **             GOLONG =MFER
493             **
494             ** History:
495             **
496             **    Date      Programmer              Modification
497             **  --------   ----------   ----------------------------------
498             **  11/09/83   MB           Documentation
499             **
500             ********************************************************************
501             ********************************************************************
502 0AE2D 3100 =NORDIM LC(2)  =eVCNTX        CAN'T REDIMENSION
503 0AE31 66F1  GLORP  GOTO   SMFERR
504 0AE35 692F  BIGLNK GOTO   TOOBIG
505             ********************************************************************
506             ********************************************************************
507             **
508             ** Name:(S) DMNSN   -  Create And Allocate Memory For Variable
509             **
510             ** Category:   VARMGT
511             **
512             ** Purpose:
513             **      Create simple numeric/string variable, numeric array
514             **      and string vector.
515             **
516             ** Entry:
517             **      Array(S0)  = 1 Create array
518             **                 = 0 Create simple variable
519             **      String(S1) = 1 String variable
520             **                 = 0 Numeric variable
521             **      NonEx(S2)  = 1 Create new variable
522             **                 = 0 Redimension existing array
523             **      D = Dope vector of the variable
524             **      A = # of elements of the array
525             **      C = Element length in nibbles
526             **      D0 = PC
527             **      R2(X) = Variable name
528             **      S-R1-1 = Variable address if already exist
529             **
```

```
530                ** Exit:
531                **      Carry CLEAR if PC is pointing at end of line.
532                **
533                ** Calls:      A-MULT, CR-VAR, CR-ARR, AJDEST, ARYSIZ, CR-ADJ,
534                **             ADRS40, WIPOUT
535                **
536                ** Uses:       A,B,C,D,R0,R1,R2,R3,S3,P
537                **
538                ** Stk lvls:   3
539                **
540                ** History:
541                **
542                **    Date     Programmer          Modification
543                ** --------   ----------   ----------------------------------
544                **             SA           Wrote
545                **
546                ****************************************************************
547                ****************************************************************
548 0AE39 853  =DMNSN  ST=1   NoZero         Don't zero array contents
549 0AE3C 136         CDOEX                 SAVE PC ON HARDWARE STACK
550 0AE3F 06          RSTK=C
551 0AE41 136         CDOEX
552 0AE44 1F00        D1=(5) =S-R1-2
            000
553 0AE4B 147         C=DAT1 A              GET VARIABLE NAME
554 0AE4E 10A         R2=C
555 0AE51 872         ?ST=1  NonEx          NEW VARIABLE?
556 0AE54 23          GOYES  DMNSN1         IF SO, GOTO DMNSN1
557 0AE56 D5          B=C    A              NEW
558 0AE58 8E00        GOSUBL =ADRS40        NEW; SEARCH FOR VARIABLE
            00
559 0AE5E D9          C=B    A              NEW
560 0AE60 135         D1=C
561 0AE63 1537        A=DAT1 W              READ PRIMARY VARIABLE
562 0AE67 310E        LCHEX  E0
563 0AE6B 9E6         ?A>C   B              PARAMETER?
564 0AE6E FB          GOYES  NORDIM         IF SO, GOTO NORDIM
565 0AE70 871         ?ST=1  String         STRING?
566 0AE73 65          GOYES  REDIM          IF SO, GOTO REDIM
567 0AE75 31A0        LCHEX  0A
568 0AE79 982         ?A<C   P              SIMPLE REAL?
569 0AE7C A0          GOYES  DMNSN1         IF SO GOTO DMNSN1
570 0AE7E 30B         LCHEX  B
571 0AE81 9E6         ?A>C   B              SIMPLE INTEGER OR SHORT ?
572 0AE84 54          GOYES  REDIM          IF NOT, GOTO REDIM
573 0AE86 118  DMNSN1 C=R0                  C= REQUIRED SIZE
574 0AE89 8E00        GOSUBL =chkspc        CHECK IF MEMORY SUFFICIENT
            00
575 0AE8F 45A         GOC    BIGLNK         CARRY CLEAR IF MEMORY SUFFICIENT
576 0AE92 112         A=R2
577 0AE95 AB8         B=A    X              RETRIEVE NAME
578 0AE98 7F14        GOSUB  CR-VAR         CREATE PRIMARY VARIABLE
579 0AE9C 119         C=R1
580 0AE9F 1547        DAT0=C W              STORE PRIMARY VARIABLE
581 0AEA3 860         ?ST=0  Array          ARRAY?
```

```
582 0AEA6 A0          GOYES  DMNSN2          IF NOT, GOTO DMNSN2
583 0AEA8 7295        GOSUB  CR-ARR          CREATE ARRAY SPACE
584 0AEAC 7E05        GOSUB  AJDEST          ADJUST DESTINATION ADDRESS
585 0AEB0 07   DMNSN2 C=RSTK                 RESTORE THE PC
586 0AEB2 134         D0=C
587 0AEB5 14A  ENDCHK A=DATO B               READ NEXT TOKEN
588 0AEB8 161         D0=D0+ 2
589 0AEBB 3100        LC(2)  =tCOMMA
590 0AEBF 962         ?A=C   B               COMMA?
591 0AEC2 00          RTNYES                 IF SO, RETURN CARRY SET
592 0AEC4 181         D0=D0- 2
593 0AEC7 03          RTNCC                  RETURN CARRY CLEAR
594
595 0AEC9 119  REDIM  C=R1
596 0AECC 861         ?ST=0  String          NUMERIC?
597 0AECF A0          GOYES  REDIM1          IF SO, GOTO REDIM1
598 0AED1 AB0         A=0    X
599 0AED4 AB2         C=0    X
600 0AED7 26          P=     6
601 0AED9 912  REDIM1 ?A=C   WP              WIPEOUT REQUIRED?
602 0AEDC 50          GOYES  REDIM2          IF NOT, GOTO REDIM2
603 0AEDE 843         ST=0   NoZero          SET FOR WIPEOUT
604 0AEE1 20   REDIM2 P=     0
605            * CREATE THE NEW VARIABLE OVER THE EXISTING VARIABLE
606            *
607 0AEE3 8E23        GOSUBL ARYSIZ          COMPUTE EXISTING ARRAY SIZE
          70
608            *
609            * NOW D1 POINTS TO THE ARRAY POINTER
610            *     C = ARRAY POINTER ( OFFSET RELATIVE TO D1)
611            *     A = EXISTING ARRAY SIZE IN NIBBLES
612            *
613 0AEE9 D7          D=C    A               D = ARRAY POINTER
614 0AEEB 137         CD1EX
615 0AEEE 06          RSTK=C                 SAVE D1 ON RSTK
616 0AEF0 D8          B=A    A               SAVE A(A) IN B(A)
617            *
618            * Check if any pending dest is in the range of this array ?
619            * If so, don't let them redimension this array
620            *
621 0AEF2 7874        GOSUB  getcnt
622 0AEF6 463         GOC    CRV230
623 0AEF9 7874        GOSUB  PRMCHN
624 0AEFD 17E  CRV210 D1=D1+ 15
625 0AF00 17E         D1=D1+ 15
626 0AF03 174         D1=D1+ 5
627 0AF06 7144        GOSUB  DESTCK
628 0AF0A 5B1         GONC   CRV220
629 0AF0D 07          C=RSTK
630 0AF0F 06          RSTK=C
631 0AF11 EB          C=C-D  A               C(A) = low addr of the array
632 0AF13 143         A=DAT1 A
633 0AF16 8B2         ?A<C   A
634 0AF19 D0          GOYES  CRV220
635 0AF1B C9          C=C+B  A
```

```
 636 0AF1D 8BE          ?A>=C  A
 637 0AF20 60           GOYES  CRV220
 638               *
 639 0AF22 6A0F         GOTO   NORDIM          CAN'T REDIMENSION
 640               *
 641               * See if we get to the bottom of the nested user-def fn yet ?
 642               *
 643 0AF26 7274  CRV220 GOSUB  NXPRM+
 644 0AF2A 42D          GOC    CRV210
 645               *
 646 0AF2D 07    CRV230 C=RSTK                 Restore D1 & A(A)
 647 0AF2F 135          D1=C
 648 0AF32 D4           A=B    A
 649 0AF34 11A          C=R2                   RESTORE VARIABLE NAME TO B(X)
 650 0AF37 AB5          B=C    X
 651 0AF3A 128          CROEX                 C = NEW ARRAY SIZE
 652 0AF3D 8B2          ?C>A   A              NEW > EXISTING ?
 653 0AF40 51           GOYES  CRV240         IF SO, GOTO CRV240
 654 0AF42 8B6          ?C<A   A              NEW < EXISTING ?
 655 0AF45 D4           GOYES  CRV260         IF SO, GOTO CRV260
 656               * NEW AND EXISTING ARRAY ARE THE SAME SIZE
 657 0AF47 1CA          D1=D1- 11
 658 0AF4A 129          CR1EX                 C= THE DOPE VECTOR
 659 0AF4D 15DA         DAT1=C 11
 660 0AF51 6570         GOTO   CRV300
 661               * NEW ARRAY LARGER THAN EXISTING ARRAY, EXPAND EXISTING ARRAY
 662 0AF55 100   CRV240 R0=A                  R0= EXISTING ARRAY SIZE
 663 0AF58 E2           C=C-A  A              C= ADDITIONAL SPACE
 664 0AF5A DF           CDEX   A              D= ADDITIONAL, C=ARRAY POINTER
 665 0AF5C EE           C=A-C  A              EXISTING ARRAY END ADDR = D1-C+A
 666 0AF5E 133          AD1EX
 667 0AF61 CA           A=A+C  A
 668 0AF63 130          D0=A                  D0 POINTS EXISTING ARRAY END
 669 0AF66 DB           C=D    A              C= ADDITIONAL SPACE REQUIRED
 670 0AF68 8E00         GOSUBL =chkspc        CHECK IF STILL ROOM
           00
 671 0AF6E 560          GONC   CRV250         IF NOT, SIGNAL NO ROOM
 672 0AF71 6DED         GOTO   TOOBIG
 673 0AF75 DB    CRV250 C=D    A              C= ADDITIONAL SPACE TO EXPAND
 674 0AF77 7E75         GOSUB  GAP
 675 0AF7B 78A5         GOSUB  ADJPTR         ADJUST ALL THE CHAIN POINTERS
 676 0AF7F 136          CDOEX                 D0 POINTS TO NEXT CHAIN HEAD
 677 0AF82 DA           A=C    A              A= LOW END ADDR OF NEW GAP
 678 0AF84 DB           C=D    A
 679 0AF86 AC2          C=0    S              DENOTE POSITIVE ADJUSTMENT
 680 0AF89 128          CROEX                 R0= ADDITIONAL, C= EXISTING SIZE
 681 0AF8C EA           A=A-C  A              A= NEW ARRAY BASE ADDRESS
 682 0AF8E 6030         GOTO   CRV270
 683               * NEW ARRAY SMALLER THAN EXISTING ARRAY, SHRINK EXISTING ARRAY
 684 0AF92 EA    CRV260 A=A-C  A              A= # OF NIBBLES TO EXTRACT
 685 0AF94 137          CD1EX                 D1= NEW ARRAY SIZE
 686 0AF97 EB           C=C-D  A              C= START OF ARRAY ADDRESS
 687 0AF99 DE           ACEX   A              C= # OF NIBBLES, A= START ADDRESS
 688 0AF9B D7           D=C    A              D= # OF NIBBLES TO EXTRACT
 689 0AF9D C2           C=A+C  A              C= NEW ARRAY START ADDRESS
```

```
690 0AF9F 108          RO=C                RO= NEW ARRAY START ADDRESS
691 0AFA2 137          CD1EX               C= NEW ARRAY SIZE
692 0AFA5 C2           C=A+C  A            C= START EXTRACTING POINT
693 0AFA7 134          DO=C
694 0AFAA 7925         GOSUB  EXTRCT
695 0AFAE FB           D=-D   A            D= 10'S CMPLMT OF NIBS TO EXTRACT
696 0AFB0 7375         GOSUB  ADJPTR       ADJUST ALL CHAIN POINTERS
697 0AFB4 136          CDOEX               DO POINTS TO NEXT CHN HEAD ADDR
698 0AFB7 120          AROEX               A= NEW ARRAY START ADDRESS
699 0AFBA DB           C=D    A
700 0AFBC 108          RO=C                RO= 10'S CMPLMT OF EXTRACTED SIZE
701 0AFBF 71C4 CRV270  GOSUB  CR-ADJ       ADJ ARRAY PTRS IN THE SAME CLASS
702 0AFC3 77F3         GOSUB  AJDEST       ADJUST DEST.ADDR
703 0AFC7 11A  CRV300  C=R2                GET THE VARIABLE NAME BACK
704 0AFCA AB5          B=C    X
705 0AFCD 8E00         GOSUBL =ADRS40      FIND THE VARIABLE ADDRESS AGAIN
          00
706 0AFD3 870          ?ST=1  Array        CREATING ARRAY VARIABLE ?
707 0AFD6 21           GOYES  CRV310       IF SO, GOTO CRV310
708 0AFD8 871          ?ST=1  String       IS IT A STRING VARIABLE ?
709 0AFDB D0           GOYES  CRV310       IF SO, GOTO CRV310
710 0AFDD 119          C=R1
711 0AFE0 1547         DAT0=C W            ZERO THE THE NUMBER
712 0AFE4 6BCE         GOTO   DMNSN2       GOTO DMNSN2
713 0AFE8 873  CRV310  ?ST=1  NoZero       FOR RE DIM ?
714 0AFEB B1           GOYES  CRV320       IF SO, GOTO CRV320
715 0AFED 136          CDOEX               SET D1 = DO
716 0AFF0 135          D1=C
717 0AFF3 7426         GOSUB  ARYSIZ       COMPUTE ITS NEW SIZE
718 0AFF7 133          AD1EX               A= ARRAY PTR ADDR, D1= ARRAY SIZE
719 0AFFA EE           C=A-C  A            C= ARRAY START ADDRESS
720 0AFFC 137          CD1EX               D1= ARY START ADDR, C= ARY SIZE
721 0AFFF 8F00         GOSBVL =WIPOUT      ZERO THE ENTIRE ARRAY
          000
722 0B006 69AE CRV320  GOTO   DMNSN2       GOTO DMNSN2
723               ****************************************************************
724               ****************************************************************
725               **
726               ** Name:    STAT    -  STAT Statement Execute
727               **
728               ** Category:   STEXEC
729               **
730               ** Purpose:
731               **      Execute STAT statement.
732               **
733               ** Entry:
734               **      DO=PC.
735               **      Jumped on STAT token.
736               **
737               ** Exit:
738               **      Errors out (eARGOR) if arg out of range.
739               **      Through NXTSTM.
740               **
741               ** Calls:      Cslc5, DMNSN, DPVCTR, PREP, SPACE.
742               **
```

```
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------  ----------------------------------
                **             SA          Wrote
                **  10/18/83   NM          Attempted to document
                **
                ********************************************************************
                ********************************************************************
0B00A 00000             REL(5) =STATDC
      0
0B00F 00000             REL(5) =STATP
      0
0B014 779D =STAT        GOSUB  PREP
0B018 870               ?ST=1  Array         NEW STAT ARRAY?
0B01B 05                GOYES  STAT40        IF SO  GOTO STAT40
0B01D D4                A=B    A
0B01F 862               ?ST=0  NonEx         SUPPOSEDLY EXISTING ARRAY FOUND?
0B022 B1                GOYES  STAT20        IF SO, GOTO STAT20
0B024 3100 =NOSTAT      LC(2)  =eIVSAR       NO SUCH STAT ARRAY
0B028 8C00  SMFERR      GOLONG =MFERR        ERROR
      00
0B02E 31CE  STAT10      LCHEX  EC
0B032 966               ?A#C   B             PARAMETER POINTER?
0B035 FE                GOYES  NOSTAT        IF NOT, GOTO NOSTAT
0B037 171               D1=D1+ 2
0B03A 143               A=DAT1 A             READ PARAMETER ADDRESS
0B03D 131   STAT20      D1=A
0B040 143               A=DAT1 A             READ TYPE INFORMATION
0B043 32C1              LCHEX  81C
      8
0B048 936               ?A#C   X             STAT ARRAY?
0B04B 3E                GOYES  STAT10        IF NOT, REPEAT STAT10
0B04D 1C2               D1=D1- 3
0B050 1573              C=DAT1 X
0B054 1F00  STAT30      D1=(5) =STATAR
      000
0B05B 1553              DAT1=C X             STORE STAT ARRAY NAME IN STATAR
0B05F 8C00  CRTRTN      GOLONG =NXTSTM       RETURN
      00
0B065 8C00  ARGOR       GOLONG =ARGERR
      00
0B06B 71EB  STAT40      GOSUB  DPVCTR        GENERATE DOPE VECTOR
0B06F D2                C=0    A
0B071 30F               LCHEX  F
0B074 8B3               ?C<D   A             MORE THAN 15 INDEPENDENT VARS
0B077 EE                GOYES  ARGOR         IF SO, GOTO ARGOR
0B079 DB                C=D    A
0B07B D5                B=C    A
0B07D C9    STAT50      C=B+C  A             COMPUTE NUMBER OF ARRAY ELEMENTS
0B07F CD                B=B-1  A
0B081 5BF               GONC   STAT50
```

```
791 0B084 DA          A=C    A
792 0B086 E4          A=A+1  A              NUMBER OF ELEMENTS IN A(A)
793 0B088 7000        GOSUB  =Cslc5         DIMLIMIT IN NIBBLES 7-10
794 0B08C DB          C=D    A              INDEPENDENT VARIABLE COUNT IN 3-6
795 0B08E F2          CSL    A
796 0B090 BF2         CSL    W
797 0B093 BF2         CSL    W
798 0B096 32C1        LCHEX  81C            STAT WITH BASE 0, DIM 1, REAL
          8
799 0B09B 109         R1=C                  R1=DOPE VECTOR
800 0B09E 7BFC        GOSUB  SPACE          COMPUTE SPACE REQUIREMENTS
801 0B0A2 739D        GOSUB  DMNSN          CREATE ARRAY
802 0B0A6 11A         C=R2                  LOAD THE STAT VARIABLE NAME
803 0B0A9 6AAF        GOTO   STAT30         GOTO STAT30
804             ********************************************************************
805             ********************************************************************
806             **
807             ** Name:    VCTRCK  -  Check Dope Vector For Simple Var
808             **
809             ** Category:   VARMGT
810             **
811             ** Purpose:
812             **      Determine if a dope vector contains a simple variable
813             **      or is a pointer to something.
814             **
815             ** Entry:
816             **      P=0.
817             **      A[B]=first byte of dope vector.
818             **
819             ** Exit:
820             **      P=0.
821             **      Carry set iff dope vector contains simple variable
822             **        (i.e., simple integer or short variable).
823             **
824             ** Calls:      None.
825             **
826             ** Uses.......
827             **             C[B].
828             **
829             ** Stk lvls:   0
830             **
831             ** History:
832             **
833             **    Date     Programmer             Modification
834             **  --------   ----------   ---------------------------------
835             **             SA           Wrote
836             **  10/18/83   NM           Attempted to document
837             **
838             ********************************************************************
839             ********************************************************************
840 0B0AD 31A0  VCTRCK LCHEX  0A
841 0B0B1 982         ?A<C   P
842 0B0B4 00          RTNYES
843 0B0B6 B66         C=C+1  B              C(B)= 0B
844 0B0B9 9EA         ?A<=C  B              IS INTEGER OR SHORT ?
```

```
 845 0B0BC 00          RTNYES
 846 0B0BE 310E        LCHEX  E0
 847 0B0C2 9EE         ?A>=C  B              IS AN INDIRECT REFERENCE ?
 848 0B0C5 00          RTNYES
 849 0B0C7 01          RTN
 850             *
 851             ********************************************************************
 852             ********************************************************************
 853             **
 854             ** Name:    DSTRY*  -  Destroy A Variable
 855             **
 856             ** Category:   VARMGT
 857             **
 858             ** Purpose:
 859             **      Destroy a variable.
 860             **
 861             ** Entry:
 862             **      D0=PC, pointing at a tokenized variable name.
 863             **      S0 = 1 iff we are in a user-defined function.
 864             **        This will prevent destruction of global variables
 865             **        from within a user-defined function.
 866             **
 867             ** Exit:
 868             **      D0 pointing past variable tokenization.
 869             **      Variable has been destroyed.
 870             **
 871             ** Calls:       ADDRSS, ADJMFG, ADJPTR, ARYSIZ, C=ACTV, EXTR10,
 872             **              EXTRCT, PRMC20, VCTRCK, getcnt.
 873             **
 874             ** Uses.......
 875             **              A,B,C,D,D0,D1,P.
 876             **
 877             ** Stk lvls:   3
 878             **
 879             ** History:
 880             **
 881             **    Date      Programmer              Modification
 882             **  --------    ----------   ---------------------------------
 883             **              SA           Wrote
 884             **  10/18/83    NM           Attempted to document
 885             **
 886             ********************************************************************
 887             ********************************************************************
 888 0B0C9 8E00 =DSTRY* GOSUBL =ADDRSS        SEARCH FOR VARIABLE
         00
 889 0B0CF D6          C=A    A
 890 0B0D1 132         ADOEX
 891 0B0D4 400         RTNC                  IF NOT FOUND, RETURN CARRY SET
 892 0B0D7 06          RSTK=C                SAVE PC ON HARDWARE STACK
 893 0B0D9 817         DSRC                  D(B)=# OF VAR LEFT IN SAME CLASS
 894 0B0DC 817         DSRC                  SAVE POINTER IN D(15,14)
 895 0B0DF 130         D0=A                  SEE IF A LOCAL OR GLOBAL VARIABLE
 896 0B0E2 100         R0=A                  R0= VARIABLE ADDRESS
 897 0B0E5 182         D0=D0- 3
 898 0B0E8 1563        C=DAT0 X              READ THE VARIABLE NAME
```

```
899 0B0EC 109          R1=C                 R1= VARIABLE NAME
900 0B0EF 7DA1         GOSUB  C=ACTV
901 0B0F3 8B2          ?A<C   A
902 0B0F6 B0           GOYES  DSTR30        IF A LOCAL VAR, GOTO DSTR30
903 0B0F8 860          ?ST=0  InDEF         ARE WE IN A LOCAL ENVIRONMENT ?
904 0B0FB 60           GOYES  DSTR30        IF NOT, GOTO DSTR30
905 0B0FD 67F0 NODSTR  GOTO   DSTR90        DON'T DESTROY GLOBAL IN LOCAL ENV
906 0B101 130  DSTR30  D0=A
907 0B104 D3           D=0    A
908 0B106 14A          A=DAT0 B
909 0B109 70AF         GOSUB  VCTRCK        SEE IF IS A NUMBER OR VECTOR
910 0B10D 402          GOC    DSTR35        IF IS A NUMBER, GOTO DSTR35
911            * THE VARIABLE IS AN ARRAY, STRING, OR COMPLEX NUMBER
912 0B110 136          CD0EX
913 0B113 135          D1=C                 D1= VARIABLE ADDRESS
914 0B116 8EFF         GOSUBL ARYSIZ        COMPUTE ARRAY SIZE
          40
915 0B11C DE           ACEX   A             A=ARRAY POINTER, C= ARRAY SIZE
916 0B11E D7           D=C    A             D=ARRAY SIZE
917 0B120 137          CD1EX                C= ARRAY POINTER ADDRESS
918 0B123 E2           C=C-A  A
919 0B125 134          D0=C                 D0= ARRAY START SDDRESS
920 0B128 8E9A         GOSUBL EXTRCT        EXTRACT THE ARRAY
          30
921 0B12E D2   DSTR35  C=0    A
922 0B130 3131         LC(2)  19
923 0B134 C3           D=D+C  A
924 0B136 118          C=R0
925 0B139 135          D1=C                 D1=VARIABLE ADDRESS
926 0B13C AFB          C=D    W
927 0B13F 812          CSLC
928 0B142 812          CSLC                 C(B)= # OF VARS BELOW THIS VAR
929 0B145 D5           B=C    A
930 0B147 860          ?ST=0  InDEF         IS A LOCAL VARIABLE ?
931 0B14A 40           GOYES  DSTR36        IF NOT, GOTO DSTR36
932 0B14C E5           B=B+1  A             INCLUDES THE FUNCTION VALUE TOO
933 0B14E 17A  DSTR36  D1=D1+ 11            POINT TO NEXT VARIABLE
934 0B151 177  DSTR37  D1=D1+ 8
935 0B154 A6D          B=B-1  B             DONE WITH VARS IN SAME CLASS ?
936 0B157 4A1          GOC    DSTR40
937 0B15A 14B          A=DAT1 B
938 0B15D 7C4F         GOSUB  VCTRCK        SEE IF IT IS A VECTOR
939 0B161 4CE          GOC    DSTR36        IF NOT, GOTO TO SEE NEXT VARIABLE
940 0B164 17A          D1=D1+ 11            POINT TO THE PTR OF THE VECTOR
941 0B167 147          C=DAT1 A
942 0B16A EB           C=C-D  A             ADJUST IT
943 0B16C 145          DAT1=C A
944 0B16F 51E          GONC   DSTR37        (B.E.T.)
945 0B172 119  DSTR40  C=R1
946 0B175 D5           B=C    A             B(X) = PARAMETER NAME
947 0B177 860          ?ST=0  InDEF         IS THIS A FCN PARAMETER ?
948 0B17A 63           GOYES  DSTR70        IF NOT, GOTO DSTR70
949 0B17C 1F00         D1=(5) =MTHSTK       ADJUST MTHSTK,FORSTK & GSBSTK
          000
950 0B183 7570         GOSUB  ADJMFG
```

```
                * WE HAVE TO ADJUST OFFSETS TO PREVIOUS MTHSTK, FORSTK & GSBSTK
                * THAT ARE SAVED ON THE FUNCTION STACK
                *
0B187 137               CD1EX
0B18A 1C2               D1=D1- 3
0B18D 1CF               D1=D1- 16               D1 @ TOP OF GSBSTK
0B190 7EE1              GOSUB  PRMC20
0B194 D2                C=0    A
0B196 3184              LC(2)  (ufMSTK)-(ufRTNA)
0B19A 133               AD1EX
0B19D CA                A=A+C  A
0B19F 131               D1=A                    D1 @ OFFSET TO PREVIOUS MTHSTK
0B1A2 FB                D=-D   A
0B1A4 7450              GOSUB  ADJMFG
0B1A8 72C1              GOSUB  getcnt           ADJUST PARM COUNT & CHAIN ADDRESS
                *
0B1AC 6310              GOTO   DSTR80
                *
0B1B0 FB        DSTR70  D=-D   A
0B1B2 8EF6              GOSUBL ADJPTR
              30
0B1B8 1C1               D1=D1- 2
0B1BB 147               C=DAT1 A
0B1BE CE                C=C-1  A              DECREMENT VARIABLES NUMBER
                *
0B1C0 14D       DSTR80  DAT1=C B
0B1C3 171               D1=D1+ 2
0B1C6 D2                C=0    A
0B1C8 3131              LC(2)  19
0B1CC D7                D=C    A
0B1CE 147               C=DAT1 A
0B1D1 CB                C=C+D  A
0B1D3 145               DAT1=C A
0B1D6 118               C=RO
0B1D9 134               DO=C
0B1DC 182               DO=DO- 3                DO @ VARIABLE NAME
0B1DF 1F00              D1=(5) =MTHSTK          RAISE MTHSTK BY 19 NIBBLES
              000
0B1E6 143               A=DAT1 A
0B1E9 D2                C=0    A
0B1EB 3131              LC(2)  19
0B1EF EE                C=A-C  A
0B1F1 7CE2              GOSUB  EXTR10
0B1F5 07        DSTR90  C=RSTK
0B1F7 134               DO=C                    RESTORE PC
0B1FA 02                RTNSC                   RETURN
                *
                *
0B1FC 2D        ADJMFG  P=     13
0B1FE 147       ADJ=PS  C=DAT1 A
0B201 CB                C=C+D  A
0B203 145               DAT1=C A
0B206 174               D1=D1+ 5
0B209 0C                P=P+1
0B20B 52F               GONC   ADJ=PS
```

```
1004 0B20E 01          RTN
1005            ********************************************************************
1006            ********************************************************************
1007            **
1008            ** Name:    DSTROY  -  DESTROY Statement Execute
1009            **
1010            ** Category:   STEXEC
1011            **
1012            ** Purpose:
1013            **      DESTROY statement execute.
1014            **
1015            ** Entry:
1016            **      Jumped on DESTROY token.
1017            **      D0=PC.
1018            **
1019            ** Exit:
1020            **      Through NXTSTM.
1021            **
1022            ** Calls:      A=CALS, ADJ=PS, C=ACTV, DSTRY*, ENDCHK, EXTRCT,
1023            **             getcnt.
1024            **
1025            ** NOTE:
1026            **      The DESTROY command has only local meaning.  If the
1027            **      machine is in a local environment, such as DEF FN or
1028            **      subprogram, only local variables can be destroyed.
1029            **      If DESTROY ALL is executed in a DEF FN, it will be
1030            **      treated as END DEF.
1031            **
1032            ** Detail:
1033            **      DESTROY ALL  |  DESTROY <var>,<var>,<var>,...
1034            **
1035            ** History:
1036            **
1037            **    Date     Programmer          Modification
1038            **  --------   ----------   ----------------------------------
1039            **             SA           Wrote
1040            **  10/18/83   NM           Attempted to document
1041            **
1042            ********************************************************************
1043            ********************************************************************
1044 0B210 0000         REL(5) =DSTRDC
           0
1045 0B215 0000         REL(5) =DSTRYP
           0
1046 0B21A 1F00 =DSTROY D1=(5) =FORSTK        SET MTHSTK=FORSTK
           000
1047 0B221 147          C=DAT1 A
1048 0B224 1C4          D1=D1- 5
1049 0B227 145          DAT1=C A
1050 0B22A 840          ST=0   InDEF
1051 0B22D 7D31         GOSUB  getcnt         SEE IF WE ARE IN A DEF FN
1052 0B231 450          GOC    DSTR05
1053 0B234 850          ST=1   InDEF          REMEMBER IN DEF FN
1054 0B237 3100  DSTR05 LC(2)  =tALL          SEE IF IT IS DESTROY ALL
1055 0B23B 14A          A=DAT0 B
```

```
1056 0B23E 966          ?A#C   B
1057 0B241 B6           GOYES  DSTR10
1058            * DESTROY ALL
1059 0B243 860   DSTALL ?ST=0  InDEF          IN A DEF FN ?
1060 0B246 80           GOYES  DSTA30         IF NOT, GOTO DSTA30
1061 0B248 8C00         GOLONG =DEFEND
           00
1062            * DESTROY ALL GLOBAL VARIABLES
1063 0B24E 7240 =DSTA30 GOSUB  A=CALS         DESTROY ALL ABOVE CALL STACK
1064 0B252 BF0          ASL    W
1065 0B255 BF0          ASL    W
1066 0B258 1B00         DO=(5) (=CHNLST)+175
           000
1067 0B25F 3191         LC(2)  25             INIT ENTIRE CHAIN HEAD TABLE
1068 0B263 1586  DSTA50 DAT0=A 7
1069 0B267 186          DO=DO- 7
1070 0B26A A6E          C=C-1  B
1071 0B26D 55F          GONC   DSTA50
1072 0B270 7020         GOSUB  A=CALS         A= CALSTK
1073 0B274 7820         GOSUB  C=ACTV         C= ACTIVE
1074 0B278 134          DO=C
1075 0B27B EE           C=A-C  A
1076 0B27D D7           D=C    A
1077 0B27F 7452         GOSUB  EXTRCT         EXTRACT ALL THE VARIABLES
1078 0B283 1F00         D1=(5) =MTHSTK        ADJ MTHSTK,FORSTK,GSBSTK,ACTIVE
           000
1079 0B28A 2C           P=     12
1080 0B28C 7E6F         GOSUB  ADJ=PS
1081 0B290 6ECD         GOTO   CRTRTN
1082            *
1083 0B294 1F00  A=CALS D1=(5) =CALSTK
           000
1084 0B29B 143          A=DAT1 A
1085 0B29E 01           RTN
1086 0B2A0 1F00  C=ACTV D1=(5) =ACTIVE
           000
1087 0B2A7 147          C=DAT1 A
1088 0B2AA 01           RTN
1089
1090            *
1091 0B2AC 791E  DSTR10 GOSUB  DSTRY*
1092 0B2B0 710C         GOSUB  ENDCHK         END OF LINE?
1093 0B2B4 47F          GOC    DSTR10         IF NOT, REPEAT DSTR10
1094 0B2B7 67AD         GOTO   CRTRTN
1095            ********************************************************************
1096            ********************************************************************
1097            **
1098            ** Name:    CR-VAR  -  Create Space For A Variable.
1099            **
1100            ** Category:   VARMGT
1101            **
1102            ** Purpose:
1103            **      Create space in the variable chain for a variable
1104            **      or a dope vector.
1105            **
```

```
1106                ** Entry:
1107                **      P=0.
1108                **      B[X] = 3-nibble variable name.
1109                **      HEX mode.
1110                **
1111                ** Exit:
1112                **      P=0.
1113                **      Carry set ->
1114                **        Variable already exists.  D0 points at variable or
1115                **          dope vector (pointing PAST variable name).
1116                **      Carry clear ->
1117                **        New space created for variable.  D0 points past name
1118                **          at area for variable or dope vector.
1119                **
1120                ** Calls:       ADJPTR, CHNHED, CR-V30, CR-VCK, DESTCK, GAP,
1121                **              NEWVAR, NXPRM+, PRMCHN, getcnt.
1122                **
1123                ** Uses.......
1124                **              A,B,C,D,P,D0,D1,R3.
1125                **
1126                ** Stk lvls:    2
1127                **
1128                ** History:
1129                **
1130                **    Date      Programmer              Modification
1131                **  --------    ----------   ----------------------------------
1132                **              SA           Wrote
1133                **  10/18/83    NM           Attempted to document
1134                **
1135                ********************************************************************
1136                ********************************************************************
1137 0B2BB 8E00 =CR-VAR GOSUBL =CHNHED        GET CHAIN HEAD POINTER
           00
1138 0B2C1 AE2          C=0     B            INITIALIZE
1139 0B2C4 AEF          CDEX    B
1140 0B2C7 AEA          A=C     B
1141 0B2CA 4A1          GOC     CR-V20       IF CHAIN IS NULL, GOTO CR-V20
1142 0B2CD 1563  CR-V10 C=DAT0  X            READ CHAIN LABEL
1143 0B2D1 162          D0=D0+  3            POINT TO REGISTER CONTENTS
1144 0B2D4 931          ?B=C    X            VARIABLE FOUND?
1145 0B2D7 00           RTNYES               IF SO, RETURN CARRY SET
1146 0B2D9 B67          D=D+1   B            INCREMENT CHAIN ELEMENT COUNTER
1147 0B2DC 16F          D0=D0+  16           POINT TO NEXT CHAIN ELEMENT
1148 0B2DF A6C          A=A-1   B            CHAIN EXHAUSTED?
1149 0B2E2 5AE          GONC    CR-V10       IF NOT, REPEAT CR-V10
1150 0B2E5 3431  CR-V20 LCHEX   00013
           000
1151 0B2EC 7902         GOSUB   GAP          ALLOCATE CHAIN REGISTER
1152 0B2F0 7332         GOSUB   ADJPTR
1153 0B2F4 7372         GOSUB   NEWVAR       UPDATE CHAIN HEAD POINTER
1154 0B2F8 AB9          C=B     X
1155 0B2FB 1543         DAT0=C  X            WRITE CHAIN LABEL
1156 0B2FF 162          D0=D0+  3            POINT TO REGISTER CONTENTS
1157 0B302 AF2          C=0     W
1158 0B305 3131         LC(2)   19
```

```
1159 0B309 10B          R3=C                    R3= ADJUST 19 NIBBLES
1160 0B30C 7E50         GOSUB  getcnt           SEE IF WE ARE IN A DEF FN ?
1161 0B310 483          GOC    CR-V45           IF NOT, RETURN CARRY CLEAR
1162                    *
1163                    * Adjust D0 to point to end of the newly created reg vector.
1164                    *
1165 0B313 16F          D0=D0+ 16
1166 0B316 7500         GOSUB  CR-V30           Adjust pointers if necessary
1167 0B31A 18F          D0=D0- 16
1168 0B31D 03           RTNCC
1169                    *
1170 0B31F 7250  CR-V30 GOSUB  PRMCHN           GET ADDR OF TOP OF FN STACK
1171 0B323 74E0  CR-V35 GOSUB  CR-VCK           SEE IF RETURN PC IS ON STACK
1172 0B327 17E   CR-V38 D1=D1+ 15
1173 0B32A 17E          D1=D1+ 15
1174 0B32D 174          D1=D1+ 5                D1 POINTS TO SAVED "STMTD1"
1175 0B330 77D0         GOSUB  CR-VCK           UPDATE IT IF NEED TO
1176                    *
1177 0B334 7310         GOSUB  DESTCK
1178 0B338 560          GONC   CR-V40
1179                    *
1180 0B33B 7CC0         GOSUB  CR-VCK
1181                    *
1182 0B33F 7950  CR-V40 GOSUB  NXPRM+
1183 0B343 4FD          GOC    CR-V35
1184 0B346 133          AD1EX
1185 0B349 03    CR-V45 RTNCC
1186                    *
1187                    ****************************************************************
1188                    *
1189 0B34B 1C4   DESTCK D1=D1- 5                D1 POINTS TO 1ST RETURN ADDRESS
1190 0B34E 143          A=DAT1 A                IF 1ST RTN ADDR SAVED IS =STORE,
1191 0B351 3400         LC(5)  =STORE            THEN MUST BE AN ASNMNT STATEMENT
          000
1192 0B358 179          D1=D1+ 10               D1 @ SAVED S-RO-O
1193 0B35B 8A2          ?A=C   A                 AND WE HAVE TO LOOK AT THE
1194 0B35E 00           RTNYES                   SAVED DESTINATION ADDRESS
1195 0B360 3400         LC(5)  =VALCHK          OR 1ST RTN ADDR SAVED IS =VALCHK
          000
1196 0B367 8A2          ?A=C   A
1197 0B36A 00           RTNYES
1198 0B36C 03           RTNCC
1199                    *
1200                    *******
1201                    *
1202 0B36E 8D00  getcnt GOVLNG =GETCNT
          000
1203                    *
1204                    ****************************************************************
1205                    ****************************************************************
1206                    **
1207                    ** Name:    PRMCHN  -  Compute Address Of Top Of Fn Stack
1208                    **
1209                    ** Category:   VARMGT
1210                    **
```

```
1211                    ** Purpose:
1212                    **      Compute address of top of FN stack.
1213                    **
1214                    ** Entry:
1215                    **      HEX mode.
1216                    **
1217                    ** Exit:
1218                    **      HEX mode.
1219                    **      D1 points at top of FN stack.
1220                    **
1221                    ** Calls:      None.
1222                    **
1223                    ** Uses.......
1224                    **              D1,C[A],C[S].
1225                    **
1226                    ** Stk lvls:   0
1227                    **
1228                    ** History:
1229                    **
1230                    **    Date      Programmer              Modification
1231                    **  --------    ----------   ----------------------------------
1232                    **              SA           Wrote
1233                    **  10/18/83    NM           Attempted to document
1234                    **
1235                    ********************************************************************
1236                    ********************************************************************
1237 0B375 1F00 =PRMCHN D1=(5) =GSBSTK        FN STACK IS RIGHT UNDER GSBSTK
           000
1238 0B37C 147          C=DAT1 A
1239 0B37F 135 =PRMC10  D1=C
1240 0B382 1574  PRMC20 C=DAT1 S              C(S)- RETURN TYPE
1241 0B386 170          D1=D1+ 1
1242 0B389 147          C=DAT1 A              LAST GSBSTK SHOULD BEEN 000000
1243 0B38C 174          D1=D1+ 5
1244 0B38F 8AE          ?C#0   A              REACH LAST GSBSTK YET ?
1245 0B392 0F           GOYES  PRMC20         IF NOT, KEEP GOING
1246 0B394 B46          C=C+1  S              RETURN TYPE SHOULD BE "F"
1247 0B397 5AE          GONC   PRMC20
1248 0B39A 01           RTN
1249                    *
1250                    ********************************************************************
1251                    *
1252 0B39C 17F   NXPRM+ D1=D1+ 16
1253 0B39F 17F          D1=D1+ 16
1254 0B3A2 17E          D1=D1+ 15
1255                    *
1256 0B3A5 14F  =NXPRMP C=DAT1 B              D1 POINTS AT PREV PRMPTR COUNT
1257 0B3A8 A6E          C=C-1  B
1258 0B3AB 540          GONC   NXPR10         IF PREV PRMPTR COUNT#0, GO NXPR10
1259 0B3AE 03           RTNCC                 NO MORE FN NESTING, CLEAR CARRY
1260 0B3B0 1C4   NXPR10 D1=D1- 5              D1 POINTS TO PREV GSBSTK OFFSET
1261 0B3B3 147          C=DAT1 A
1262 0B3B6 133          AD1EX
1263 0B3B9 C2           C=A+C  A              COMPUTE PREVIOUS GSBSTK ADDRESS
1264 0B3BB 53C          GONC   PRMC10         (B.E.T.)
```

```
1265            ****************************************************************
1266            ****************************************************************
1267            **
1268            ** Name:    AJDEST  -  Adjust Destination Address
1269            **
1270            ** Category:   VARMGT
1271            **
1272            ** Purpose:
1273            **      Adjust destination address, whatever that means.
1274            **
1275            ** Entry:
1276            **      R0=Adjustment amount.
1277            **
1278            ** Exit:
1279            **      P       =  0
1280            **
1281            ** Calls:      ARYSIZ, PRMCHN, CR-VCK, NXPRM+
1282            **
1283            ** Uses.......
1284            **              A,B,C,D,P,D0,D1,R3.
1285            **
1286            ** Stk lvls:   3
1287            **
1288            ** History:
1289            **
1290            **    Date      Programmer              Modification
1291            **   --------   ----------   ----------------------------------
1292            **              SA           Wrote
1293            **   10/18/83   NM           Attempted to document
1294            **
1295            ****************************************************************
1296            ****************************************************************
1297 0B3BE 1F00 =AJDEST D1=(5) =PRMPTR        SEE IF WE ARE IN A DEF FN
          000
1298 0B3C5 14F          C=DAT1 B
1299 0B3C8 A6E          C=C-1  B
1300 0B3CB 5D1          GONC   AJDE30         RETURN IF NOT
1301 0B3CE 118          C=R0
1302 0B3D1 10B          R3=C                  SAVE ADJUSTMENT TO R3
1303 0B3D4 D9           C=B    A
1304 0B3D6 134          D0=C                  D0 = DOPE VECTOR ADDRESS
1305 0B3D9 16A          D0=D0+ 11             POINTS TO ARRAY POINTER
1306 0B3DC 146          C=DAT0 A
1307 0B3DF 132          ADOEX
1308 0B3E2 EA           A=A-C  A
1309 0B3E4 130          D0=A                  D0 = START OF ARRAY
1310 0B3E7 03           RTNCC
1311 0B3E9 D9    AJDE30 C=B    A
1312 0B3EB 10B          R3=C                  SAVE DOPE VECTOR ADDR IN R3
1313 0B3EE 135          D1=C
1314 0B3F1 7622         GOSUB  ARYSIZ
1315 0B3F5 EE           C=A-C  A              ARRAY END ADDR = D1-C+A
1316 0B3F7 133          AD1EX
1317 0B3FA CA           A=A+C  A
1318 0B3FC 130          D0=A                  D0 POINTS TO ARRAY END
```

```
1319 0B3FF 118          C=R0
1320 0B402 12B          CR3EX                   R3= ADJUSTMENT
1321 0B405 D5           B=C    A                RESTORE B
1322 0B407 671F         GOTO   CR-V30
1323              ****************************************************************
1324 0B40B 143   CR-VCK A=DAT1 A
1325 0B40E 136          CDOEX
1326 0B411 134          DO=C
1327 0B414 8BE          ?A>=C  A                HAS THE ADDRESS BEEN AFFECTED ?
1328 0B417 00           RTNYES                  RETURN IF NOT
1329 0B419 3400         LC(5)  =AVMEMS
          000
1330 0B420 133          AD1EX                   SEE IF THE ADDR IS ON THE STACK
1331 0B423 135          D1=C
1332 0B426 147          C=DAT1 A
1333 0B429 133          AD1EX
1334 0B42C 143          A=DAT1 A
1335 0B42F 8B2          ?A<C   A                IS THE ADDR >= AVMEMS ?
1336 0B432 00           RTNYES                  IF NOT, RETURN
1337 0B434 11B          C=R3
1338 0B437 EA           A=A-C  A
1339 0B439 141          DAT1=A A
1340 0B43C 03           RTNCC
1341              *
1342              ****************************************************************
1343              ****************************************************************
1344              **
1345              ** Name:    CR-ARR  -  Create Space For An Array
1346              **
1347              ** Category:   VARMGT
1348              **
1349              ** Purpose:
1350              **      Create space for an array.
1351              **
1352              ** Entry:
1353              **      Variable creation data in D, DO,and R0.
1354              **      P       =  0
1355              **
1356              ** Exit:
1357              **      P       =  0
1358              **
1359              ** Calls:      ADJPTR, FIND, GAP.
1360              **
1361              ** Uses.......
1362              **  Exclusive: A(A),C,D(B),DO,D1
1363              **  Inclusive: A,B,C,D(A),DO,D1
1364              **
1365              ** Stk lvls:   2
1366              **
1367              ** History:
1368              **
1369              **    Date      Programmer              Modification
1370              **  --------    ----------   ----------------------------------
1371              **              SA           Wrote
1372              **  10/28/83    NM           Attempted to document
```

```
1373                    **  12/16/83   FH          Added more documentation
1374                    **
1375                    ********************************************************************
1376                    ********************************************************************
1377 0B43E 31A0 =CR-ARR LCHEX  0A              INITIALIZE
1378 0B442 182   CR-A10 D0=D0- 3
1379 0B445 A6F          D=D-1  B               BACKCHAIN EXHAUSTED?
1380 0B448 4B2          GOC    CR-A20          IF SO, GOTO CR-A20
1381 0B44B 18F          D0=D0- 16
1382 0B44E 14A          A=DAT0 B               READ TYPE INFORMATION
1383 0B451 982          ?A<C   P               REAL REGISTER?
1384 0B454 EE           GOYES  CR-A10          IF SO, REPEAT CR-A10
1385 0B456 30D          LCHEX  D
1386 0B459 9E2          ?A<C   B               RESIDENT NON-REAL?
1387 0B45C 2E           GOYES  CR-ARR          IF SO, REPEAT CR-ARR
1388 0B45E BE4          ASR    B
1389 0B461 986          ?A>C   P               PARAMETER POINTER?
1390 0B464 AD           GOYES  CR-ARR          IF SO, REPEAT CR-ARR
1391 0B466 16A          D0=D0+ 11
1392 0B469 142          A=DAT0 A               READ RELATIVE ADDRESS
1393 0B46C 136          CDOEX
1394 0B46F E2           C=C-A  A               COMPUTE ARRAY BASE ADDRESS
1395 0B471 134          D0=C
1396 0B474 118   CR-A20 C=R0
1397 0B477 7E70         GOSUB  GAP             CREATE ARRAY SPACE
1398 0B47B 78A0         GOSUB  ADJPTR
1399 0B47F 136          CDOEX                  LOCATE REGISTER
1400 0B482 DA           A=C    A
1401 0B484 1C1   CR-ADJ D1=D1- 2
1402 0B487 14F          C=DAT1 B
1403 0B48A AE7          D=C    B
1404 0B48D A6F          D=D-1  B
1405 0B490 8E00         GOSUBL =FIND
          00
1406 0B496 119   CR-A30 C=R1
1407 0B499 15CA         DAT0=C 11              STORE DOPE VECTOR
1408 0B49D 16A          D0=D0+ 11
1409 0B4A0 EC           A=B-A  A               COMPUTE ARRAY RELATIVE ADDRESS
1410 0B4A2 34B0         LCHEX  0000B
          000
1411 0B4A9 6A10         GOTO   CR-A50          GOTO CR-A50
1412 0B4AD 16F   CR-A40 D0=D0+ 16              POINT TO NEXT REGISTER
1413 0B4B0 162          D0=D0+ 3
1414 0B4B3 14A          A=DAT0 B               READ TYPE INFORMATION
1415 0B4B6 982          ?A<C   P               REAL REGISTER?
1416 0B4B9 31           GOYES  CR-A60          IF SO, GOTO CR-A60
1417 0B4BB 118          C=R0
1418 0B4BE 16A          D0=D0+ 11
1419 0B4C1 142          A=DAT0 A
1420 0B4C4 CA    CR-A50 A=A+C  A               UPDATE ARRAY POINTER
1421 0B4C6 140          DAT0=A A
1422 0B4C9 18A          D0=D0- 11
1423 0B4CC 30A   CR-A60 LCHEX  A
1424 0B4CF A6F          D=D-1  B               CHAIN EXHAUSTED?
1425 0B4D2 5AD          GONC   CR-A40          IF NOT, REPEAT CR-A40
```

```
1426 0B4D5 02          RTNSC                  RETURN CARRY SET
1427           ********************************************************************
1428           * EXTRCT - EXTRACT A BLOCK MEMORY
1429           *   INPUT:  D = # OF NIBBLES TO EXTRACT
1430           *           D0  POINTS TO START EXTRACTING POINT
1431           *
1432 0B4D7 1F00  EXTRCT D1=(5) =MTHSTK
          000
1433 0B4DE 147         C=DAT1 A               C= MTHSTK
1434 0B4E1 132  EXTR10 ADOEX                  A= START EXTRACTING POINT
1435 0B4E4 EE          C=A-C  A               C= STACK SIZE
1436 0B4E6 130         D0=A                   D0= START EXTRACTING POINT
1437 0B4E9 DF          CDEX   A               C= # OF NIBBLES TO EXTRACT
1438 0B4EB CA          A=A+C  A               A= END EXTRACTING POINT
1439 0B4ED 131         D1=A                   D1= END EXTRACTING POINT
1440 0B4F0 DF          CDEX   A
1441 0B4F2 8D00 =MOved3 GOVLNG =MOVED3
          000
1442           ********************************************************************
1443           *
1444           *  GAP  -  Create gap for variable and zero it out
1445           *
1446           *  Uses:  A,C,D(A),D0,D1
1447           *
1448 0B4F9 D7   GAP    D=C    A               SAVE GAP LENGTH
1449 0B4FB 1F00        D1=(5) =MTHSTK
          000
1450 0B502 143         A=DAT1 A
1451 0B505 EE          C=A-C  A
1452 0B507 135         D1=C
1453 0B50A 136         CDOEX
1454 0B50D 8F00        GOSBVL =MOVEU2         OPEN GAP
          000
1455 0B514 DB          C=D    A
1456 0B516 8F00        GOSBVL =WIPOUT         ZERO GAP
          000
1457 0B51D 136         CDOEX
1458 0B520 EB          C=C-D  A               SET D0 TO START OF GAP
1459 0B522 134         D0=C
1460 0B525 03          RTNCC                  RETURN CARRY CLEAR
1461           ********************************************************************
1462           ********************************************************************
1463           **
1464           ** Name:    ADJPTR  -  Adjust Stack And Chain Pointers
1465           **
1466           ** Category:   PTRUTL
1467           **
1468           ** Purpose:
1469           **      Adjust stack and chain pointers.
1470           **
1471           ** Entry:
1472           **      HEX mode.
1473           **      D[A] = amount to SUBTRACT from pointers.
1474           **      P=0.
1475           **      B[B]=ASCII code of highest parmchain head to adjust
```

```
1476            **       (e.g., B[B]=\Z\).
1477            **
1478            ** Exit:
1479            **      P      = 0.
1480            **      Carry clear.
1481            **
1482            ** Calls:      None.
1483            **
1484            ** Uses.......
1485            **             A[B],C[A],P,D1.
1486            **
1487            ** Stk lvls:   0
1488            **
1489            ** Detail:
1490            **      Adjusts MTHSTK, FORSTK, GSBSTK, ACTIVE and all parm
1491            **      chain heads from A to whatever was specified in B[B].
1492            **
1493            ** History:
1494            **
1495            **    Date     Programmer              Modification
1496            **  --------   ----------   ----------------------------------
1497            **             SA           Wrote
1498            **  10/18/83   NM           Attempted to document
1499            **
1500            ****************************************************************
1501            ****************************************************************
1502 0B527 1F00 =ADJPTR D1=(5) =MTHSTK       ADJUST STACK & CHAIN POINTERS
           000
1503 0B52E 2C           P=     12
1504 0B530 147   AP-10  C=DAT1 A             ADJUST STACK POINTERS
1505 0B533 EB           C=C-D  A
1506 0B535 145          DAT1=C A
1507 0B538 174          D1=D1+ 5
1508 0B53B 0C           P=P+1
1509 0B53D 52F          GONC   AP-10
1510 0B540 17B          D1=D1+ 12            D1 POINTS TO (=PRMPTR+2)
1511 0B543 AE4          A=B    B
1512 0B546 31F5         LCHEX  5F
1513 0B54A 0E66         A=A&C  B
1514 0B54E 3114         LCHEX  41
1515 0B552 B6A          A=A-C  B
1516 0B555 147          C=DAT1 A             READ PARAMETER CHAIN ADDRESS
1517 0B558 EB    AP-20  C=C-D  A             ADJUST CHAIN HEAD ADDRESS
1518 0B55A 145          DAT1=C A
1519 0B55D 176          D1=D1+ 7
1520 0B560 147          C=DAT1 A             READ NEXT CHAIN HEAD ADDRESS
1521 0B563 A6C          A=A-1  B             MORE ADDRESSES TO ADJUST?
1522 0B566 51F          GONC   AP-20         IF SO, REPEAT AP-20
1523 0B569 03           RTNCC                RETURN CARRY CLEAR
1524            ****************************************************************
1525 0B56B EB    NEWVAR C=C-D  A             ADJUST CHAIN HEAD ADDRESS
1526 0B56D 145          DAT1=C A
1527 0B570 1C1          D1=D1- 2
1528 0B573 14F          C=DAT1 B
1529 0B576 AE7          D=C    B
```

```
1530 0B579 B66          C=C+1  B              INCREMENT CHAIN COUNTER
1531 0B57C 14D          DAT1=C B
1532 0B57F 01           RTN                   RETURN
1533              *
1534              ********************************************************************
1535              ********************************************************************
1536              **
1537              ** Name:(S) DATLEN  -  Compute Data Length Given Type
1538              **
1539              ** Category:   VARMGT
1540              **
1541              ** Purpose:
1542              **      Compute length of a data item.
1543              **
1544              ** Entry:
1545              **      C[0]=data type.
1546              **        5 - Integer.
1547              **        4 - Short real.
1548              **        3 - Real.
1549              **        2 - Short complex.
1550              **        1 - Complex.
1551              **      P=0.
1552              **
1553              ** Exit:
1554              **      C[A]=Length of data item:
1555              **        Integer: 6.
1556              **        Short real: 9.
1557              **        Real: 10H.
1558              **        Short complex: 12H.
1559              **        Complex: 20H.
1560              **
1561              ** Calls:      None.
1562              **
1563              ** Uses.......
1564              **             C.
1565              **
1566              ** Stk lvls:   0
1567              **
1568              ** History:
1569              **
1570              **    Date     Programmer             Modification
1571              **  --------   ----------   ----------------------------------
1572              **             SA           Wrote
1573              **  10/18/83   NM           Attempted to document
1574              **
1575              ********************************************************************
1576              ********************************************************************
1577 0B581 B8E   LENGTH C=-C-1 P
1578 0B584 A06  =DATLEN C=C+C  P
1579 0B587 B8A          C=-C   P
1580 0B58A 80D0         P=C    0
1581 0B58E 3B20         LCHEX  060910122002
           0221
           0190
           60
```

```
1582 0B59C 22          P=     2
1583 0B59E 3200        LCHEX  000
          0
1584 0B5A3 20          P=     0
1585 0B5A5 01          RTN
1586            ****************************************************************
1587            ****************************************************************
1588            **
1589            ** Name:(S) ARYSIZ  -  Compute Array Size, # Elements
1590            ** Name:(S) ARYELM  -  Compute Array Size, # Elements
1591            **
1592            ** Category:   VARMGT
1593            **
1594            ** Purpose:
1595            **   ARYSIZ: Compute array size in bytes.
1596            **   ARYELM: Compute number of elements in an array.
1597            **
1598            ** Entry:
1599            **       D1 points at the dope vector of the array.
1600            **
1601            ** Exit:
1602            **       P=0.
1603            **   ARYELM: D1 points at first subscript limit.
1604            **           A = number of elements in the array.
1605            **   ARYSIZ: D1 points at the array pointer within the array
1606            **             dope vector.
1607            **           C = array pointer (is an offset from the array
1608            **             pointer to the start of the array).
1609            **           A = array size in nibbles.
1610            **
1611            ** Calls:       ARYELM: A-MULT.
1612            **               ARYSIZ: ARYELM, DATLEN, A-MULT.
1613            **
1614            ** Uses.......
1615            **               A,B,C,D,D1.
1616            **
1617            ** Stk lvls:    ARYELM: 1.
1618            **               ARYSIZ: 2.
1619            **
1620            ** History:
1621            **
1622            **    Date      Programmer              Modification
1623            **  --------    ----------   ----------------------------------
1624            **                SA           Wrote
1625            **  10/18/83      NM           Attempted to document
1626            **
1627            ****************************************************************
1628            ****************************************************************
1629            *
1630 0B5A7 1573 =ARYELM C=DAT1 X
1631 0B5AB AB7         D=C    X
1632 0B5AE A27         D=D+D  XS            PUSH OUT THE STATISTIC BIT
1633 0B5B1 A27         D=D+D  XS
1634 0B5B4 A27         D=D+D  XS
1635 0B5B7 20          P=     0
```

```
1636 0B5B9 30F          LCHEX  F
1637 0B5BC B0F          D=C-D  P            D(0)= 0-5 FOR ORGINAL D(0)= F-A
1638 0B5BF 172          D1=D1+ 3            D1 @ 2ND SUBSCR LIM OR MAX STR LEN
1639 0B5C2 D2           C=0    A
1640 0B5C4 15F3         C=DAT1 4            C=2ND SUBSCR LIM OR MAX STR LEN
1641 0B5C8 173          D1=D1+ 4            D1 @ 1ST SUBSCRIPT LIMIT
1642 0B5CB D0           A=0    A
1643 0B5CD 15B3         A=DAT1 4            A=1ST SUBSCRIPT LIMIT
1644 0B5D1 92F          ?D#0   XS           SEE WHAT IS THE OPTION BASE
1645 0B5D4 B0           GOYES  ARYS10
1646 0B5D6 E4           A=A+1  A            OPTION BASE 0
1647 0B5D8 90B          ?D=0   P            IS THIS A STRING ?
1648 0B5DB 02           GOYES  ARYS20
1649 0B5DD E6           C=C+1  A
1650 0B5DF 90B  ARYS10  ?D=0   P            IS THIS A STRING ?
1651 0B5E2 91           GOYES  ARYS20       YES, IF D(0) ORIGINALLY = F
1652 0B5E4 21           P=     1            D(1)= SUBSCRIPT COUNT
1653 0B5E6 A0F          D=D-1  P            IF D(1)=0 IS NOT ARRAY OR VECTOR
1654 0B5E9 590          GONC   ARYS15       IF IS AN ARRAY, GOTO ARYS15
1655 0B5EC D0           A=0    A
1656 0B5EE E4           A=A+1  A
1657 0B5F0 462          GOC    ARYS50       (B.E.T.)
1658 0B5F3 90F   ARYS15 ?D#0   P            IF IS A 2 SUBSCRIPTS ARRAY,
1659 0B5F6 B1           GOYES  ARYS40       IF SO, MULTIPLY THEM
1660 0B5F8 5E1          GONC   ARYS50       (B.E.T.)
1661 0B5FB DE   ARYS20  ACEX   A
1662            * NOW A=MAX. STRING LENGTH,  C= 1ST SUBSCRIPT LIMIT
1663 0B5FD E4           A=A+1  A            ADD 2 BYTES FOR STR HDR(LENGTH)
1664 0B5FF E4           A=A+1  A
1665 0B601 C4           A=A+A  A            A= STRING LENGTH IN NIBBLES
1666 0B603 21           P=     1
1667 0B605 90F          ?D#0   P            IS THIS A STRING VECTOR ?
1668 0B608 90           GOYES  ARYS40
1669 0B60A D2   ARYS30  C=0    A            NOT ARRAY NOR STRING, IS COMPLEX
1670 0B60C E6           C=C+1  A
1671 0B60E 580          GONC   ARYS50       (B.E.T.)
1672 0B611 8E00 ARYS40  GOSUBL =a-mult      MULTIPLY THE ROW & COLUM LIMIT
          00
1673 0B617 20   ARYS50  P=     0
1674 0B619 02           RTNSC
1675            *
1676 0B61B 788F =ARYSIZ GOSUB  ARYELM
1677            *
1678 0B61F 90B  ARYS55  ?D=0   P            IS THIS A STRING ?
1679 0B622 F0           GOYES  ARYS60
1680 0B624 A8B          C=D    P
1681 0B627 795F         GOSUB  DATLEN       COMPUTE DATA LENGTH
1682 0B62B 8E00         GOSUBL =a-mult      COMPUTE THE ARRAY LENGTH
          00
1683 0B631 173   ARYS60 D1=D1+ 4            POINT TO RELATIVE POINTER
1684 0B634 147          C=DAT1 A
1685 0B637 03           RTNCC
1686            *
1687            *
1688            *
```

```
1689                *
1690 0B639                 END
```

```
 A=CALS  Abs   45716 #0B294 -  1083  1063  1072
 ACTIVE  Ext               -  1086
 ADDRSS  Ext               -   434   888
 ADJ=PS  Abs   45566 #0B1FE -   998  1003  1080
 ADJMFG  Abs   45564 #0B1FC -   997   950   964
=ADJPTR  Abs   46375 #0B527 -  1502   675   696   970  1152  1398
 ADRS40  Ext               -   558   705
 AJDE30  Abs   46057 #0B3E9 -  1311  1300
=AJDEST  Abs   46014 #0B3BE -  1297   584   702
 AP-10   Abs   46384 #0B530 -  1504  1509
 AP-20   Abs   46424 #0B558 -  1517  1522
 ARGERR  Ext               -   779
 ARGOR   Abs   45157 #0B065 -   779   329   785
 ARGORL  Abs   44441 #0AD99 -   329   319   323
=ARYELM  Abs   46503 #0B5A7 -  1630  1676
 ARYS10  Abs   46559 #0B5DF -  1650  1645
 ARYS15  Abs   46579 #0B5F3 -  1658  1654
 ARYS20  Abs   46587 #0B5FB -  1661  1648  1651
 ARYS30  Abs   46602 #0B60A -  1669
 ARYS40  Abs   46609 #0B611 -  1672  1659  1668
 ARYS50  Abs   46615 #0B617 -  1673  1657  1660  1671
 ARYS55  Abs   46623 #0B61F -  1678
 ARYS60  Abs   46641 #0B631 -  1683  1679
=ARYSIZ  Abs   46619 #0B61B -  1676   607   717   914  1314
 AVMEMS  Ext               -  1329
 Array   Abs       0 #00000 -    13   188   240   420   425   581   706   755
 BASE    Ext               -   453
 BIGLNK  Abs   44597 #0AE35 -   504   575
 C=ACTV  Abs   45728 #0B2A0 -  1086   439   900  1073
 CALSTK  Ext               -  1083
 CHNHED  Ext               -  1137
 CHNLST  Ext               -  1066
 CNFLCT  Ext               -   276
 COLLAP  Ext               -   117
 CR-A10  Abs   46146 #0B442 -  1378  1384
 CR-A20  Abs   46196 #0B474 -  1396  1380
 CR-A30  Abs   46230 #0B496 -  1406
 CR-A40  Abs   46253 #0B4AD -  1412  1425
 CR-A50  Abs   46276 #0B4C4 -  1420  1411
 CR-A60  Abs   46284 #0B4CC -  1423  1416
 CR-ADJ  Abs   46212 #0B484 -  1401   701
=CR-ARR  Abs   46142 #0B43E -  1377   583  1387  1390
 CR-V10  Abs   45773 #0B2CD -  1142  1149
 CR-V20  Abs   45797 #0B2E5 -  1150  1141
 CR-V30  Abs   45855 #0B31F -  1170  1166  1322
 CR-V35  Abs   45859 #0B323 -  1171  1183
 CR-V38  Abs   45863 #0B327 -  1172
 CR-V40  Abs   45887 #0B33F -  1182  1178
 CR-V45  Abs   45897 #0B349 -  1185  1161
=CR-VAR  Abs   45755 #0B2BB -  1137   578
 CR-VCK  Abs   46091 #0B40B -  1324  1171  1175  1180
 CRTRTN  Abs   45151 #0B05F -   777    81  1081  1094
 CRV210  Abs   44797 #0AEFD -   624   644
 CRV220  Abs   44838 #0AF26 -   643   628   634   637
 CRV230  Abs   44845 #0AF2D -   646   622
```

```
 CRV240   Abs   44885 #0AF55 -   662   653
 CRV250   Abs   44917 #0AF75 -   673   671
 CRV260   Abs   44946 #0AF92 -   684   655
 CRV270   Abs   44991 #0AFBF -   701   682
 CRV300   Abs   44999 #0AFC7 -   703   660
 CRV310   Abs   45032 #0AFE8 -   713   707   709
 CRV320   Abs   45062 #0B006 -   722   714
 Cslc5    Ext                -   793
 DATATP   Abs   44389 #0AD65 -   276   315
=DATLEN   Abs   46468 #0B584 -  1578  1681
 DECDC    Ext                -    61    67
 DECP     Ext                -    62
 DEFEND   Ext                -  1061
 DESTCK   Abs   45899 #0B34B -  1189   627  1177
=DIM      Abs   44048 #0AC10 -    69
 DIMP     Ext                -    68
=DIMSTM   Abs   44048 #0AC10 -    70    65
=DMNSN    Abs   44601 #0AE39 -   548    79   801
 DMNSN1   Abs   44678 #0AE86 -   573   556   569
 DMNSN2   Abs   44720 #0AEB0 -   585   582   712   722
 DMSTM1   Abs   44068 #0AC24 -    75    80
=DPVCTR   Abs   44112 #0AC50 -   187    76   781
=DSTA30   Abs   45646 #0B24E -  1063  1060
 DSTA50   Abs   45667 #0B263 -  1068  1071
 DSTALL   Abs   45635 #0B243 -  1059
 DSTR05   Abs   45623 #0B237 -  1054  1052
 DSTR10   Abs   45740 #0B2AC -  1091  1057  1093
 DSTR30   Abs   45313 #0B101 -   906   902   904
 DSTR35   Abs   45358 #0B12E -   921   910
 DSTR36   Abs   45390 #0B14E -   933   931   939
 DSTR37   Abs   45393 #0B151 -   934   944
 DSTR40   Abs   45426 #0B172 -   945   936
 DSTR70   Abs   45488 #0B1B0 -   969   948
 DSTR80   Abs   45504 #0B1C0 -   975   967
 DSTR90   Abs   45557 #0B1F5 -   992   905
 DSTRDC   Ext                -  1044
=DSTROY   Abs   45594 #0B21A -  1046
=DSTRY*   Abs   45257 #0B0C9 -   888  1091
 DSTRYP   Ext                -  1045
 ENDCHK   Abs   44725 #0AEB5 -   587  1092
 EXPEX+   Ext                -   118
 EXTR10   Abs   46305 #0B4E1 -  1434   991
 EXTRCT   Abs   46295 #0B4D7 -  1432   694   920  1077
 FIND     Ext                -  1405
 FORSTK   Ext                -   197  1046
 GAP      Abs   46329 #0B4F9 -  1448   674  1151  1397
 GETCNT   Ext                -  1202
=GETDIM   Abs   44395 #0AD6B -   313   193   203   222
 GLORP    Abs   44593 #0AE31 -   503
 GSBSTK   Ext                -  1237
 GTDM10   Abs   44420 #0AD84 -   320   317
 GTDM20   Abs   44430 #0AD8E -   324   321
=INTEGR   Abs   44034 #0AC02 -    63
 InDEF    Abs       0 #00000 -    14   903   930   947  1050  1053  1059
 LENGTH   Abs   46465 #0B581 -  1577   371
```

```
=LIMITS  Abs   44094 #0AC3E -   117   190   221
 MEMERR  Ext                -   275
 MFERR   Ext                -   761
 MOVED3  Ext                -  1441
 MOVED2  Ext                -  1454
=MOved3  Abs   46322 #0B4F2 -  1441
 MTHSTK  Ext                -   949   986  1078  1432  1449  1502
 NEWVAR  Abs   46443 #0B56B -  1525  1153
 NODSTR  Abs   45309 #0B0FD -   905
=NORDIM  Abs   44589 #0AE2D -   502   441   564   639
=NOSTAT  Abs   45092 #0B024 -   760   765
 NXPR10  Abs   46000 #0B3B0 -  1260  1258
 NXPRM+  Abs   45980 #0B39C -  1252   643  1182
=NXPRMP  Abs   45989 #0B3A5 -  1256
 NXTSTM  Ext                -   777
 NoZero  Abs       3 #00003 -    18   548   603   713
 NonEx   Abs       2 #00002 -    16   433   437   555   758
 OpBase  Abs       3 #00003 -    17   232   263   320   452   455
 POP1N   Ext                -   313
=PREP    Abs   44463 #0ADAF -   418    75   754
 PREP10  Abs   44488 #0ADC8 -   427   424
 PREP20  Abs   44506 #0ADDA -   433   431
 PREP30  Abs   44535 #0ADF7 -   442   436
 PREP40  Abs   44584 #0AE28 -   456   454
=PRMC10  Abs   45951 #0B37F -  1239  1264
 PRMC20  Abs   45954 #0B382 -  1240   957  1245  1247
=PRMCHN  Abs   45941 #0B375 -  1237   623  1170
 PRMPTR  Ext                -  1297
=REAL    Abs   44034 #0AC02 -    65
 REDIM   Abs   44745 #0AEC9 -   595   566   572
 REDIM1  Abs   44761 #0AED9 -   601   597
 REDIM2  Abs   44769 #0AEE1 -   604   602
 RSTST   Ext                -   119
 S-R0-1  Ext                -   223
 S-R0-3  Ext                -   206
 S-R1-2  Ext                -   442   552
 S-R1-3  Ext                -    73
=SHORT   Abs   44034 #0AC02 -    64
 SMFERR  Abs   45096 #0B028 -   761   503
=SPACE   Abs   44445 #0AD9D -   371    78   800
=STAT    Abs   45076 #0B014 -   754
 STAT10  Abs   45102 #0B02E -   763   772
 STAT20  Abs   45117 #0B03D -   768   759
 STAT30  Abs   45140 #0B054 -   775   803
 STAT40  Abs   45163 #0B06B -   781   756
 STAT50  Abs   45181 #0B07D -   788   790
 STATAR  Ext                -   775
 STATDC  Ext                -   752
 STATP   Ext                -   753
 STORE   Ext                -  1191
 String  Abs       1 #00001 -    15   213   235   238   427   432   565   596
                              708
 TOOBIG  Abs   44383 #0AD5F -   275   253   373   504   672
 VALCHK  Ext                -  1195
 VCT10   Abs   44171 #0AC8B -   206   189   200
```

VCT15	Abs	44230	#0ACC6	-	223	219			
VCT20	Abs	44240	#0ACD0	-	225	214			
VCT30	Abs	44277	#0ACF5	-	238	233	236		
VCT40	Abs	44300	#0AD0C	-	247	239			
VCT50	Abs	44323	#0AD23	-	254	251			
VCT60	Abs	44363	#0AD4B	-	267	248			
VCT70	Abs	44365	#0AD4D	-	268	271			
VCTRCK	Abs	45229	#0B0AD	-	840	909	938		
WIPOUT	Ext			-	721	1456			
a-mult	Ext			-	252	372	1672	1682	
chkspc	Ext			-	574	670			
eIVSAR	Ext			-	760				
eVCNTX	Ext			-	502				
fltdh	Ext			-	316				
getcnt	Abs	45934	#0B36E	-	1202	621	965	1051	1160
tALL	Ext			-	1054				
tARRAY	Ext			-	422				
tCOMMA	Ext			-	589				
ufMSTK	Abs	78	#0004E	-	11	959			
ufRTNA	Abs	6	#00006	-	11	959			

Input Parameters

  Source file name is AB&REG::MS

  Listing file name is AB/REG:TI:ML::-1

  Object file name is AB%REG:TI:MS::-1

```
                                  111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                        TITLE  Functions <831212.1512>
 0B639                  ABS    #0B639
                *
                *   AAA   BBBB    &     FFFFF   CCC   N   N
                *  A   A  B   B  & &    F      C   C  N   N
                *  A   A  B   B  & &    F      C      NN  N
                *  AAAAA  BBBB    &     FFFF   C      N N N
                *  A   A  B   B  & & &  F      C      N  NN
                *  A   A  B   B  &  &   F      C   C  N   N
                *  A   A  BBBB    && &  F       CCC   N   N
                *
                        RDSYMB TI%EQU::MS
                ********************************************************************
                ********************************************************************
                **
                ** Name:     INVLUT  -  Involution Operation (^)
                ** Name:     MINUS   -  Minus Operation (-)
                ** Name:     NOT     -  Not Operation (NOT)
                ** Name:     MULTPY  -  Multiply Operation (*)
                ** Name:     DIVIDE  -  Divide Operation (/)
                ** Name:     PERCNT  -  Percent Operation (%)
                ** Name:     PLUS    -  Addition Operation (+)
                ** Name:     CONCAT  -  Concatenation Operation (&)
                ** Name:     EXPON   -  EXPONENT Function
                **
                ** Category:   FNEXEC
                **
                ** Purpose:
                **      Evaluate the various binary operators.
                **
                ** Entry:
                **      Expression execution controller jumped on appropriate
                **        token.
                **      P=0.
                **      HEX mode.
                **      D0 = PC.
                **      D1 = Mathstack pointer.
                **
                **
                ** Exit:
                **      Through EXPR.
                **
                ** Calls:      Appropriate calls for each operator.
                **
                ** Uses.......
                **              Reasonable usage for functions.
                **
                ** Stk lvls:   <=4
                **
                ** Note:
                **      When operators are parsed, unlike when functions are
                **      parsed, the parameter count is not used.  The parm
                **      counts appearing before each routine are for CALC mode.
                **
                ** History:
```

```
                **
                **    Date     Programmer          Modification
                **  --------   ----------  ----------------------------------
                **             SA          Wrote
                **  10/13/83   NM          Attempted to document
                **
                ********************************************************************
                ********************************************************************
0B639 11            NIBHEX 11               Argument count range = [1,1]
0B63B 8E31 =INVLUT  GOSUBL MPOP2N
      70
0B641 447           GOC    CLNK1
0B644 8E00          GOSUBL =YX2-12
      00
0B64A 6936          GOTO   OUTRES
                ********************************************************************
0B64E 11            NIBHEX 11               Argument count range = [1,1]
0B650 78C6 =MINUS   GOSUB  POP1N
0B654 416           GOC    CLNK1
0B657 BCC           A=-A-1 S
0B65A 68B2          GOTO   D1AEXP
                ********************************************************************
0B65E 11            NIBHEX 11               Argument count range = [1,1]
0B660 7D27 =NOT     GOSUB  POP1N+
0B664 415           GOC    CLNK1
0B667 2E            P=     14
0B669 A1C           A=A-1  WP
0B66C 69F1          GOTO   BABBAJ
                ********************************************************************
0B670 11            NIBHEX 11               Argument count range = [1,1]
0B672 7ED6 =MULTPY  GOSUB  MPOP2N
0B676 4F3           GOC    CLNK1
0B679 8E00          GOSUBL =MP2-12
      00
0B67F 6406          GOTO   OUTRES
                ********************************************************************
0B683 11            NIBHEX 11               Argument count range = [1,1]
0B685 7BC6 =DIVIDE  GOSUB  MPOP2N
0B689 4C2           GOC    CLNK1
0B68C 8E00          GOSUBL =DV2-12
      00
0B692 61F5          GOTO   OUTRES
0B696 11            NIBHEX 11               Argument count range = [1,1]
0B698 78B6 =PERCNT  GOSUB  MPOP2N
0B69C 491           GOC    CLNK1
0B69F 8E00          GOSUBL =SPLTAC
      00
0B6A5 460           GOC    uMP15            carry set means non-finite oper.
0B6A8 CE            C=C-1  A                ********BUG--SEE SB!!! **********
0B6AA CE            C=C-1  A                divide finite 15 dig res by 100.
0B6AC 8E00  uMP15   GOSUBL =MP2-15
      00
0B6B2 61D5          GOTO   OUTRES
0B6B6 6965  CLNK1   GOTO   CPOLLF
                ********************************************************************
```

```
105 0B6BA 11          NIBHEX 11           Argument count range = [1,1]
106 0B6BC 7496 =PLUS  GOSUB  MPOP2N
107 0B6C0 45F         GOC    CLNK1
108 0B6C3 8E00        GOSUBL =uAD12       Fetch Rnd Mode,Split,Add, & Round
          00
109 0B6C9 6460        GOTO   TOFN4
110           ********************************************************************
111 0B6CD 11          NIBHEX 11           Argument count range = [1,1]
112 0B6CF 7566 =CONCAT GOSUB  POP1S       READ HEADER
113 0B6D3 D8          B=A    A
114 0B6D5 137         CD1EX
115 0B6D8 C9          C=C+B  A            COMPUTE ADDRESS OF NEXT STRING
116 0B6DA 135         D1=C
117 0B6DD 136         CD0EX
118 0B6E0 D7          D=C    A            SAVE PC IN D-REGISTER
119 0B6E2 15A6        A=DAT0 7            READ HEADER
120 0B6E6 7256        GOSUB  POP1S1       CHECK FOR STRING
121 0B6EA D9          C=B    A
122 0B6EC C8          B=A+B  A            COMPUTE LENGTH OF NEW STRING
123 0B6EE 7000        GOSUB  =MOved3      PACK OUT SECOND HEADER
124 0B6F2 D4          A=B    A            CONSTRUCT NEW HEADER
125 0B6F4 BF0         ASL    W
126 0B6F7 BF0         ASL    W
127 0B6FA A0C         A=A-1  P
128 0B6FD 1CF         D1=D1- 16
129 0B700 1596        DAT1=A 7
130 0B704 8CEC        GOLONG DEXPR        RESTORE PC, RETURN
          B0
131           ********************************************************************
132 0B70A 811         NIBHEX 811          Argument count range = [1,1]
133 0B70D 7C76 =EXPON GOSUB  MPOP1N
134 0B711 44A         GOC    CLNK1
135 0B714 8E00        GOSUBL =EX12        <SM%MTH>  Return Exponent(x)
          00
136 0B71A 6965        GOTO   OUTRES
137           ********************************************************************
138           ********************************************************************
139           **
140           ** Name:    CLASS   -  Execute CLASS Function
141           **
142           ** Category:   FNEXEC
143           **
144           ** Purpose:
145           **      Execute class function.
146           **
147           ** Entry:
148           **      P=0.
149           **      D0 = PC.
150           **      D1 = Mathstack pointer.
151           **      Jumped on CLASS token.
152           **
153           ** Exit:
154           **      Through EXPR.
155           **
156           ** Calls:      POP1N, CLASSA.
```

```
157                **
158                ** History:
159                **
160                **    Date     Programmer          Modification
161                ** --------   ----------   ----------------------------------
162                **              SA          Wrote
163                **  10/13/83    NM          Attempted to document
164                **
165                ********************************************************************
166                ********************************************************************
167 0B71E 811            NIBHEX 811
168 0B721 77F5 =CLASS    GOSUB  POP1N
169 0B725 409            GOC    CLNK1
170 0B728 8E00           GOSUBL =CLASSA        <SMXMTH>  Return +/-{1,2,..5}
          00
171 0B72E 61B6  TOFN4    GOTO   FN4
172                ********************************************************************
173 0B732 62E5  DATERR   GOTO   CNFLCT         "Data Type"
174                ********************************************************************
175                ********************************************************************
176                **
177                ** Name:     SMALL    -  Create Special Consts
178                ** Name:(S)  BIG      -  Create Special Consts
179                ** Name:     BIG+     -  Create Special Consts
180                ** Name:(S)  HUGE     -  Create Special Consts
181                **
182                ** Category:   MTHUTL
183                **
184                ** Purpose:
185                **      Create constants MAXREAL, INF, EPS.
186                **
187                ** Entry:
188                **
189                ** Exit:
190                **   SMALL: C[W] = EPS.
191                **          Mode unchanged.
192                **          P=14.
193                **   BIG:   C[W] =+/-9.99999999999E499 (sign preserved from
194                **                                      entry).
195                **          DEC mode.
196                **   BIG+:  C[W] = 9.99999999999E499.
197                **          DEC mode.
198                **   HUGE:  C[W] = 0999999999999F00 (infinity).
199                **
200                ** Calls:      None.
201                **
202                ** Uses.......
203                **              C.  SMALL uses P.
204                **
205                ** Stk lvls:   0
206                **
207                ** History:
208                **
209                **    Date     Programmer          Modification
210                ** --------   ----------   ----------------------------------
```

```
211                ** SA          Wrote
212                ** 10/13/83  NM          Attempted to document
213                **
214                ********************************************************************
215                ********************************************************************
216 0B736 AF2  =SMALL  C=0     W          1.0  E-499
217 0B739 2E           P=      14
218 0B73B 3410         LCHEX   50101
          105
219 0B742 02           RTNSC
220 0B744 AC2  =BIG+   C=0     S          9.99999999999  E499
221 0B747 05   =BIG    SETDEC             (Does not change P)
222 0B749 136          CDOEX
223 0B74C 1A99         D0=HEX  0499
          40
224 0B752 136          CDOEX
225 0B755 AD2          C=0     M
226 0B758 A5E          C=C-1   M            sets carry
227 0B75B 02           RTNSC
228 0B75D AF2  =HUGE   C=0     W          +Infinity
229 0B760 04   =HUGE20 SETHEX             (Does not change P)
230 0B762 A2E          C=C-1   XS           sets carry
231 0B765 05           SETDEC
232 0B767 A5E          C=C-1   M            +9.99...99F00=inf & sets carry
233 0B76A 02           RTNSC
234                ********************************************************************
235                ********************************************************************
236                **
237                ** Name:    MAXRL   -  MAXREAL Function
238                ** Name:    MINRL   -  MINREAL Function
239                ** Name:    EPS     -  EPS Function
240                ** Name:    INF     -  INF Function
241                ** Name:    NAN     -  NAN Function
242                **
243                ** Category:   FNEXEC
244                **
245                ** Purpose:
246                **      Evaluate these no-argument functions: MAXREAL, MINREAL,
247                **      EPS, INF, NAN.
248                **
249                ** Entry:
250                **      D0 = PC.
251                **      D1 = Mathstack pointer.
252                **      P=0.
253                **      Jumped on appropriate token.
254                **
255                ** Exit:
256                **      Through EXPR.
257                **
258                ** Calls:      Whatever.
259                **
260                ** History:
261                **
262                **    Date     Programmer             Modification
263                **  --------   ----------   ----------------------------------
```

```
264                **              SA              Wrote
265                **  10/13/83    NM              Attempted to document
266                **
267                ********************************************************************
268                ********************************************************************
269 0B76C 00               NIBHEX 00               Argument count range = [0,0]
270 0B76E 72DF =MAXRL      GOSUB  BIG+
271 0B772 464              GOC    FN1
272                ********************************************************************
273 0B775 00               NIBHEX 00               Argument count range = [0,0]
274 0B777 05   =MINRL      SETDEC
275 0B779 AF1              B=0    W
276 0B77C 2E               P=     14
277 0B77E B05              B=B+1  P               Mantissa=1
278 0B781 AF0              A=0    W
279 0B784 132              ADOEX
280 0B787 1B09             D0=HEX 99490           Exponent=-510
          499
281 0B78E 132              ADOEX
282 0B791 822              SB=0                   Exact.
283 0B794 821              XM=0                   No Exception.
284 0B797 73E4             GOSUB  ures12
285 0B79B 6D10             GOTO   FN1
286                ********************************************************************
287 0B79F 00               NIBHEX 00               Argument count range = [0,0]
288 0B7A1 719F =EPS        GOSUB  SMALL
289 0B7A5 431              GOC    FN1
290                ********************************************************************
291 0B7A8 00               NIBHEX 00               Argument count range = [0,0]
292 0B7AA 7FAF =INF        GOSUB  HUGE
293 0B7AE 4A0              GOC    FN1
294                ********************************************************************
295 0B7B1 00               NIBHEX 00
296 0B7B3 8E00 =NAN        GOSUBL =SIGNAN          CREATE A SIGNALING NAN.
          00
297 0B7B9 8C00  FN1        GOLONG =FNRTN1
          00
298                ********************************************************************
299 0B7BF 6064  CLNK2      GOTO   CPOLLF
300
301                ********************************************************************
302                ********************************************************************
303                **
304                ** Name:    COMPAR  -  Execute Comparison Operator
305                **
306                ** Category:   FNEXEC
307                **
308                ** Purpose:
309                **      Evaluate comparison operator.
310                **
311                ** Entry:
312                **      D0 = PC.
313                **      D1 = Mathstack pointer.
314                **      P=0.
315                **      Jumped on comparison operator token.
```

```
316                **
317                ** Exit:
318                **      Through EXPR.
319                **
320                ** Calls:      Depending on the test:
321                **             MPOP2N, POP1S, POP1S2, STRTST, uTEST.
322                **
323                ** History:
324                **
325                **    Date      Programmer              Modification
326                **  --------    ----------   ----------------------------------
327                **              SA           Wrote
328                **  10/13/83    NM           Attempted to document
329                **
330                ********************************************************************
331                ********************************************************************
332 0B7C3 11           NIBHEX 11              Argument count range = [1,1]
333 0B7C5 1564 =COMPAR C=DAT0 S               READ PREDICATE NIBBLE
334 0B7C9 160          D0=D0+ 1               INCREMENT PC
335 0B7CC AC7          D=C    S               Predicate to D (sign)
336 0B7CF 15B6         A=DAT1 7               READ STACK SIGNATURE
337 0B7D3 B04          A=A+1  P
338 0B7D6 A64          A=A+A  B
339 0B7D9 968          ?A=0   B               STRING?
340 0B7DC 02           GOYES  STRCMP          IF SO, GOTO STRCMP
341
342 0B7DE 109          R1=C                   Save Predicate.
343 0B7E1 7F65         GOSUB  MPOP2N          Fetch Arg's, Complex?
344 0B7E5 129          CR1EX                  Bring in Predicate & save C
345 0B7E8 80DF         P=C    15              PREDICATE TO POINTER
346 0B7EC 129          CR1EX                  Restore C
347
348 0B7EF 4FC          GOC    CLNK2           IF COMPLEX, POLL FOR COMPLEX ROM
349 0B7F2 8E00         GOSUBL =uTEST          COMPARE
          00
350 0B7F8 6D60         GOTO   BABBAJ          GOTO BABBAJ
351 0B7FC 7945  STRCMP GOSUB  POP1S2          ALIGN LENGTH HEADER
352 0B800 D6           C=A    A               STORE LENGTH IN C
353 0B802 137          CD1EX
354 0B805 C2           C=C+A  A               COMPUTE ADDRESS OF SECOND STRING
355 0B807 D7           D=C    A
356 0B809 137          CD1EX
357 0B80C 7825         GOSUB  POP1S           READ NEXT STRING HEADER
358 0B810 137          CD1EX
359 0B813 C2           C=C+A  A               COMPUTE ADDRESS OF FIRST STRING
360 0B815 06           RSTK=C                 SAVE ON HARDWARE STACK
361 0B817 136          CD0EX
362 0B81A DF           CDEX   A               SAVE PC IN D
363 0B81C 137          CD1EX
364 0B81F 24           P=     4               ASSUME FIRST STRING IS LONGER
365 0B821 8B6          ?A>C   A               IS FIRST STRING LONGER?
366 0B824 D0           GOYES  SC10            IF SO, GOTO SC10
367 0B826 22           P=     2               ASSUME STRINGS ARE SAME LENGTH
368 0B828 8A2          ?A=C   A               ARE STRINGS SAME LENGTH?
369 0B82B 60           GOYES  SC10            IF SO, GOTO SC10
```

```
370 0B82D 21          P=     1            FIRST STRING IS SHORTER
371 0B82F D6          C=A    A
372 0B831 80FF  SC10  CPEX   15
373 0B835 AC5         B=C    S
374 0B838 8F00        GOSBVL =STRTST      STRINGS EQUAL?
          000
375 0B83F AFD         BCEX   W
376 0B842 5F0         GONC   SC30         IF SO, GOTO SC30
377 0B845 24          P=     4            ASSUME FIRST STRING IS GREATER
378 0B847 9F0         ?A>B   W            IS FIRST STRING GREATER?
379 0B84A 40          GOYES  SC20         IF SO, GOTO SC20
380 0B84C 21          P=     1            RECORD INFERIORITY
381 0B84E 80FF  SC20  CPEX   15
382 0B852 DB    SC30  C=D    A            RESTORE PC
383 0B854 134         D0=C
384 0B857 07          C=RSTK              PLACE STACK POINTER
385 0B859 135         D1=C
386 0B85C 1CF         D1=D1- 16
387 0B85F 0E47        C=C&D  S            DETERMINE RESULT
388 0B863 BCA =BOOLE  C=-C   S
389 0B866 AF2 =BABBAJ C=0    W            ASSUME FALSE
390 0B869 570 =LOGIC  GONC   DMPRES       IF FALSE, GOTO DMPRES
391 0B86C 2E          P=     14
392 0B86E 301         LCHEX  1            CREATE LOGICAL TRUE
393 0B871 6E65 DMPRES GOTO   FN4          DUMP RESULT TO STACK
394           ********************************************************************
395           ********************************************************************
396           **
397           ** Name:    AND     -  AND Logical Operators
398           ** Name:    OR      -  OR Logical Operators
399           ** Name:    EXOR    -  EXOR Logical Operators
400           **
401           ** Category:   FNEXEC
402           **
403           ** Purpose:
404           **      Evaluate logical operators during expression execute.
405           **
406           ** Entry:
407           **      D0 = PC.
408           **      D1 = Mathstack pointer.
409           **      P=0.
410           **      HEX mode.
411           **      Jumped on appropriate token.
412           **
413           ** Exit:
414           **      Through EXPR.
415           **
416           ** Calls:      POP2N+.
417           **
418           ** History:
419           **
420           **    Date     Programmer          Modification
421           **  --------   ----------  ----------------------------------
422           **             SA          Wrote
423           **  10/13/83   NM          Attempted to document
```

```
424                **
425                ********************************************************************
426                ********************************************************************
427 0B875 11               NIBHEX 11             Argument count range = [1,1]
428 0B877 7DD4 =AND        GOSUB  POP2N+
429 0B87B 04               SETHEX
430 0B87D 491              GOC    SLARF
431 0B880 2E               P=     14
432 0B882 B9A              C=-C   WP
433 0B885 50E              GONC   BABBAJ
434 0B888 B98   LGCTST     A=-A   WP
435 0B88B 6ADF             GOTO   BABBAJ
436                ********************************************************************
437 0B88F 11               NIBHEX 11             Argument count range = [1,1]
438 0B891 73C4 =OR         GOSUB  POP2N+
439 0B895 04               SETHEX
440 0B897 4A6   SLARF      GOC    CLNK3
441 0B89A 2E               P=     14
442 0B89C 0E1E             A=A!C  WP
443 0B8A0 57E              GONC   LGCTST         GOTO LGCTST
444                ********************************************************************
445 0B8A3 11               NIBHEX 11             Argument count range = [1,1]
446 0B8A5 7FA4 =EXOR       GOSUB  POP2N+
447 0B8A9 04               SETHEX
448 0B8AB 465              GOC    CLNK3
449 0B8AE 2E               P=     14
450 0B8B0 A98              B=A    WP
451 0B8B3 A1D              B=B-1  WP             not X
452 0B8B6 450              GOC    EXOR10
453 0B8B9 A91              B=0    WP
454 0B8BC 0E11  EXOR10     B=B&C  WP             (not X) and Y
455 0B8C0 A1E              C=C-1  WP             not Y
456 0B8C3 450              GOC    EXOR20
457 0B8C6 A92              C=0    WP
458 0B8C9 0E16  EXOR20     A=A&C  WP             X and (not Y)
459 0B8CD 0E18             A=A!B  WP             ((not X) and Y)or(X and (not Y))
460 0B8D1 66BF             GOTO   LGCTST         GOTO LGCTST
461                ********************************************************************
462                ********************************************************************
463                **
464                ** Name:    SGN     -  SGN Function
465                ** Name:    ABS     -  ABS Function
466                **
467                ** Category:   FNEXEC
468                **
469                ** Purpose:
470                **      Execute SGN, ABS functions.
471                **
472                ** Entry:
473                **      P      =  0
474                **      D0 = PC.
475                **      D1 = Mathstack pointer.
476                **      Jumped on SGN or ABS token.
477                **
478                ** Exit:
```

```
479            **      Through EXPR.
480            **
481            ** Calls:      Yes.
482            **
483            ** History:
484            **
485            **    Date      Programmer              Modification
486            **  --------    ----------   ----------------------------------
487            **              SA           Wrote
488            **  10/13/83    NM           Attempted to document
489            **
490            ****************************************************************
491            ****************************************************************
492 0B8D5 811          NIBHEX 811             Argument count range = [1,1]
493 0B8D8 7044 =SGN    GOSUB  POP1N           POP ARGUMENT
494 0B8DC 452          GOC    CLNK3
495
496 0B8DF AF2          C=0    W               (SET  C#NAN)
497 0B8E2 8E00         GOSUBL =MSN15          NAN?
          00
498
499 0B8E8 580          GONC   SGN20           not NaN
500 0B8EB AF6          C=A    W               RTN THE NAN.
501 0B8EE 4F0          GOC    SGN30           "GOTO"
502
503 0B8F1 AC6  SGN20   C=A    S
504 0B8F4 2E           P=     14
505 0B8F6 918          ?A=0   WP              X=0?  (NOT INF THOUGH)
506 0B8F9 50           GOYES  SGN30
507 0B8FB 301          LCHEX  1               +/- ONE
508
509 0B8FE 627F SGN30   GOTO   DMPRES
510            ****************************************************************
511 0B902 6D13  CLNK3   GOTO   CPOLLF
512
513 0B906 811          NIBHEX 811             Argument count range = [1,1]
514 0B909 7F04 =ABS    GOSUB  POP1N
515 0B90D 44F          GOC    CLNK3
516 0B910 AC0          A=0    S
517 0B913 1517  D1AEXP DAT1=A W
518 0B917 8C00  expr   GOLONG =EXPR
          00
519
520            ****************************************************************
521            ****************************************************************
522            **
523            ** Name:    RES     -  RES Function
524            **
525            ** Category:   FNEXEC
526            **
527            ** Purpose:
528            **      Evaluate RES function.
529            **
530            ** Entry:
531            **      P=0.
```

```
532                **      DO = PC.
533                **      D1 = Mathstack pointer.
534                **      Jumped on RES token.
535                **
536                ** Exit:
537                **      Through EXPR.
538                **
539                ** Calls:      POP1N.
540                **
541                ** History:
542                **
543                **    Date     Programmer              Modification
544                **  --------   ----------   ----------------------------------
545                **             SA           Wrote
546                **  10/13/83   NM           Attempted to document
547                **
548                ********************************************************************
549                ********************************************************************
550 0B91D 00           NIBHEX 00             Argument count range = [0,0]
551 0B91F 137  =RES    CD1EX
552 0B922 1F00         D1=(5) =RESREG
          000
553 0B929 7FE3         GOSUB  POP1N
554 0B92D 135          D1=C
555 0B930 AF6          C=A    W
556 0B933 4EC          GOC    CLNK3
557 0B936 6115         GOTO   fn1
558                ********************************************************************
559 0B93A 162   REGNUM DO=DO+ 3
560 0B93D 21           P=     1
561 0B93F A0F          D=D-1  P
562 0B942 90F          ?D#0   P              DIMCOUNT EQL 2
563 0B945 33           GOYES  RGNM4          THEN GOTO RGNM4;
564 0B947 163          DO=DO+ 4              POINT AT NEXT DIMLIMIT;
565 0B94A 7EC3  RGNM1  GOSUB  POP1N          POP SUBSCRIPT;
566 0B94E 400          RTNC                  COMPLEX THEN RETURN CARRY SET;
567 0B951 7554         GOSUB  fltdh          CONVERT TO HEX;
568 0B955 470          GOC    RGNM2          CONVERSION OK THEN GOTO RGNM2;
569 0B958 8AC          ?A#0   A              CONVERSION NOT OK
570 0B95B 00           RTNYES                THEN RETURN CARRY SET;
571 0B95D D2    RGNM2  C=0    A
572 0B95F 15E3         C=DATO 4              READ DIMLIMIT;
573 0B963 163          DO=DO+ 4
574 0B966 8B6          ?A>C   A              SUBSCRIPT OUT OF RANGE
575 0B969 00           RTNYES                THEN RETURN CARRY SET;
576 0B96B 92F          ?D#0   XS             BASEOPTION ONE
577 0B96E 60           GOYES  RGNM3          THEN GOTO RGNM3;
578 0B970 E6           C=C+1  A              INCREMENT DIMLIMIT;
579 0B972 01           RTN                   OVERFLOW THEN SET CARRY; RETURN;
580 0B974 CC    RGNM3  A=A-1  A              DECREMENT SUBSCRIPT;
581 0B976 01           RTN                   ZERO SUBSCR THEN SET CRY; RETURN;
582
583 0B978 7ECF  RGNM4  GOSUB  RGNM1          GET SECOND SUBSCRIPT;
584 0B97C 422          GOC    RGNM5          COMPLX/OUT OF RNG THEN GOTO RGNM5
585 0B97F 17F          D1=D1+ 16
```

```
586 0B982 100          R0=A                SAVE SECOND SUBSCRIPT;
587 0B985 10B          R3=C                SAVE SECOND DIMLIMIT;
588 0B988 7EBF         GOSUB  RGNM1        GET FIRST SUBSCRIPT;
589 0B98C 400          RTNC                CMPLX/OUT OF RANGE THEN GO RGNM5
590 0B98F 11B          C=R3                RECALL SECOND DIMLIMIT;
591 0B992 8E00         GOSUBL =a-mult      FIRST SUBSCRIPT * 2ND DIMLIMIT;
          00
592 0B998 118          C=R0                RECALL SECOND SUBSCRIPT;
593 0B99B CA           A=A+C  A            COMPUTE REGISTER NUMBER;
594 0B99D 03           RTNCC              RETURN CARRY CLEAR;
595
596 0B99F 17F  RGNM5   D1=D1+ 16
597 0B9A2 7673         GOSUB  POP1N        POP SECOND SUBSCRIPT;
598 0B9A6 04           SETHEX
599 0B9A8 D0           A=0    A
600 0B9AA CC           A=A-1  A            CREATE LARGE REGISTER NUMBER;
601 0B9AC 02           RTNSC              RETURN CARRY SET.
602
603 0B9AE CA   2xRGNM  A=A+C  A
604 0B9B0 146  ARYADR  C=DAT0 A            READ REL ADDRESS OF ARRAY START
605 0B9B3 E2           C=C-A  A            COMPUTE REL ADDRESS OF REGISTER
606 0B9B5 132          ADOEX
607 0B9B8 EA           A=A-C  A            COMPUTE ABS ADDRESS OF REGISTER
608 0B9BA D8           B=A    A            LEAVE ADDRESS IN B AND D0
609 0B9BC 130          D0=A
610 0B9BF 111          A=R1                LEAVE PC IN A
611 0B9C2 E4           A=A+1  A            INCREMENT PC PAST SUBSCRIPT COUNT
612 0B9C4 01           RTN                 RETURN
613
614            *VCCK0  ?C#D   P            DOPE VECTOR AT INDIRECT ADDRESS?
615            *       RTNYES              IF NOT, RETURN CARRY SET
616 0B9C6 161   VCCK1  D0=D0+ 2
617 0B9C9 146          C=DAT0 A            READ INDIRECT ADDRESS
618 0B9CC 134          D0=C
619 0B9CF 20           P=     0
620 0B9D1 1563 VECCHK  C=DAT0 X            READ REGISTER TYPE INFORMATION
621 0B9D5 AB7          D=C    X
622 0B9D8 31AE         LCHEX  EA
623 0B9DC 987          ?C>D   P            SIMPLE VARIABLE?
624 0B9DF 00           RTNYES              IF SO, RETURN CARRY SET
625 0B9E1 21           P=     1
626 0B9E3 98B          ?C<=D  P            PARAMETER POINTER?
627 0B9E6 0E           GOYES  VCCK1        IF SO, REPEAT VCCK1
628 0B9E8 132          ADOEX
629 0B9EB 1560         C=DAT0 P            READ SUBSCRIPT COUNT
630 0B9EF 132          ADOEX
631 0B9F2 907          ?C#D   P            SAME ARGUMENT COUNT?
632 0B9F5 00           RTNYES              IF NOT, RETURN CARRY SET
633 0B9F7 03           RTNCC              RETURN CARRY CLEAR
634
635 0B9F9 20   NEWVEC  P=     0
636 0B9FB 3102         LCHEX  20
637 0B9FF ABD          BCEX   X
638 0BA02 10A          R2=C                SAVE VARIABLE NAME IN R3
639 0BA05 0E65         C=B&C  B            ISOLATE STRING BIT
```

```
640                *****************
641                *       ?C=0    B            BUGFIX 9605 ON 830803 -SA
642                *       GOYES   NWVC1
643                *       LCHEX   F
644                *****************
645 0BA09 96E             ?C#0    B            STRING VARIABLE?
646 0BA0C 50              GOYES   NWVC1        IF NOT, GOTO NWVC1
647 0BA0E 30D             LCHEX   D
648 0BA11 A0E    NWVC1    C=C-1   P
649 0BA14 8E00            GOSUBL  =BASE        SET BASE OPTION
          00
650                *****************
651 0BA1A 130             D0=A
652 0BA1D 21              P=      1
653 0BA1F 1560            C=DAT0  P            READ SUBSCRIPT COUNT
654 0BA23 1B00            D0=(5)  =F-R1-3
          000
655 0BA2A 1540            DAT0=C  P            SAVE COUNT FOR ASSIGNMENT
656 0BA2E AB7             D=C     X
657 0BA31 01              RTN                  RETURN
658
659 0BA33 1B00   SAVSUB   D0=(5)  =F-R1-0
          000
660 0BA3A 140             DAT0=A  A            SAVE REG NUMBER FOR ASSIGNMENT
661 0BA3D 11A             C=R2
662 0BA40 AD2             C=0     M
663 0BA43 2F              P=      15
664 0BA45 308             LCHEX   8
665 0BA48 AF5             B=C     W
666                *****************
667                *       C=0     W
668                *       P=      0            BUGFIX 9605 ON 830803 -SA
669                *       ?D=0    P
670                *       RTNYES
671                *****************
672 0BA4B 20              P=      0
673 0BA4D A8B             C=D     P
674 0BA50 B06             C=C+1   P
675 0BA53 AF2             C=0     W
676 0BA56 500             RTNNC
677                *****************
678 0BA59 2A              P=      10
679 0BA5B 35EF            LCHEX   CFFFFE
          FFFC
680 0BA63 AB9             C=B     X
681 0BA66 18F             D0=D0-  16
682 0BA69 1547            DAT0=C  W
683 0BA6D AF5             B=C     W
684 0BA70 8E00            GOSUBL  =csrc5
          00
685                *****************
686                *       C=A     A
687                *       GOSBVL  =CSRC4       BUGFIX 9605 ON 830803 -SA
688                *       P=      0            This code labelled =B9605B,
689                *       LCHEX   000008F      and moved to a Romfix area.
```

```
690                *       RTNSC
691                **************
692 0BA76 8D00             GOVLNG =B9605B
          000
693                **************
694
695 0BA7D 787F   UNDECL    GOSUB  NEWVEC        BUILD LOWER 3 NIBS OF DOPE VECTOR
696 0BA81 1B00             D0=(5) (=DIM10)-3
          000
697 0BA88 7EAE             GOSUB  REGNUM        COMPUTE REGISTER NUMBER
698 0BA8C 4F0              GOC    RTNNAN        IF OUT OF RANGE, GOTO RTNNAN
699 0BA8F 70AF             GOSUB  SAVSUB        SAVE SUBSCRIPT INFORMATION
700 0BA93 111              A=R1
701 0BA96 E4               A=A+1  A
702 0BA98 6E90             GOTO   ARYRTN        GOTO ARYRTN
703
704                **************
705                *RTNNAN ST=0   0
706                *       D=D+1  P             BUGFIX 9605 ON 830803 -SA
707                *       GONC   RTNAN1
708                *       ST=1   0
709                **************
710 0BA9C A8B    RTNNAN    C=D    P
711 0BA9F A0E              C=C-1  P
712 0BAA2 0A               ST=C
713                **************
714 0BAA4 3100             LC(2)  =eSUBSC       LOAD ERROR NUMBER
715 0BAA8 7CC1             GOSUB  INVRES        CREATE NAN, CHECK TRAPS
716 0BAAC 04               SETHEX
717                **************
718                *       B=0    A             BUGFIX 9605 ON 830803 -SA
719                **************
720 0BAAE AF1              B=0    W
721                **************
722 0BAB1 111              A=R1                 LEAVE PC IN A(A)
723 0BAB4 E4               A=A+1  A
724                **************
725                *       ?ST=0  0
726                *       GOYES  ARYRTN        BUGFIX 9605 ON 830803 -SA
727                **************
728 0BAB6 862              ?ST=0  2             NON-COMPLEX NUMERIC REFERENCE?
729 0BAB9 E7               GOYES  ARYRTN        IF SO, GOTO ARYRTN
730 0BABB A4D              B=B-1  S
731 0BABE A4D              B=B-1  S
732 0BAC1 861              ?ST=0  1             COMPLEX NUMERIC REFERENCE?
733 0BAC4 37               GOYES  ARYRTN        IF SO, GOTO ARYRTN
734                **************
735 0BAC6 D8               B=A    A             PREPARE FOR STR HDR CONSTRUCTION
736 0BAC8 33FF             LCHEX  FFFF
          FF
737 0BACE D3               D=0    A
738 0BAD0 17B              D1=D1+ 12
739 0BAD3 8C00             GOLONG =FAKE         GOTO FAKE
          00
740
```

```
741             ****************************************
742             *BUGFIX 9613 ON 830803 -SA
743             *In all routines from REGNUM down through DMARRY, scratch
744             *registers have been rearranged with respect to TI%RM5:
745             *       R0 changed to R1
746             *       R1 changed to R2
747             *       R2 changed to R0
748             *       R3 remains    R3
749             ****************************************
750
751             ************************************************************
752             ************************************************************
753             **
754             ** Name:    ARRAY   -  Evaluate Array Reference
755             **
756             ** Category:   FNEXEC
757             **
758             ** Purpose:
759             **      Evaluates an array reference during expression execute.
760             **
761             ** Entry:
762             **      D0 = PC.
763             **      D1 = Mathstack pointer.
764             **      Jumped on tARRAY token.
765             **      P=0.
766             **
767             ** Exit:
768             **      Value of array element on top of stack.
769             **      Through FNRTN3.
770             **
771             ** Calls:      ADDRSS, VECCHK, REGNUM, ARYADR.
772             **
773             ** Detail:
774             **      Array reference is tokenized as:
775             **        expression
776             **        (expression, if 2-dimensional array)
777             **        tARRAY
778             **        variable
779             **
780             ** History:
781             **
782             **    Date     Programmer             Modification
783             **  --------   ----------   ----------------------------------
784             **             SA           Wrote
785             **  10/13/83   NM           Attempted to document
786             **
787             ************************************************************
788             ************************************************************
789 0BAD9 8812          NIBHEX 8812            Argument count range = [1,2]
790 0BADD 8E00 =ARRAY   GOSUBL =ADDRSS         LOOK FOR ARRAY DOPE VECTOR
        00
791 0BAE3 101           R1=A                   SAVE PC IN R1
792 0BAE6 469           GOC    UNDECL          IF NOT FOUND, GOTO UNDECL
793 0BAE9 D1            B=0    A
794 0BAEB AB5           B=C    X               SAVE NAME IN B(A)
```

```
795 0BAEE 7FDE          GOSUB  VECCHK          CHECK OUT DOPE VECTOR
796 0BAF2 4A6           GOC    DOOFUS          IF NOT OK, GOTO DOOFUS
797 0BAF5 A27           D=D+D  XS              KNOCK OUT STATISTICS STATUS
798 0BAF8 A27           D=D+D  XS
799 0BAFB A27           D=D+D  XS
800 0BAFE 783E          GOSUB  REGNUM          COMPUTE REGISTER NUMBER
801 0BB02 136           CDOEX
802 0BB05 1B00          D0=(5) =F-R1-0
          000
803 0BB0C 140           DAT0=A A               SAVE REGISTER NUMBER FOR TRACE
804 0BB0F 136           CDOEX
805 0BB12 498           GOC    RTNNAN          IF OUT OF RANGE, GOTO RTNNAN
806 0BB15 AC1           B=0    S               INITIALIZE STORAGE TYPE INDICATOR
807 0BB18 B07           D=D+1  P               STRING?
808 0BB1B 454           GOC    STRING          IF SO, GOTO STRING
809 0BB1E 30D           LCHEX  D
810 0BB21 B0F           D=C-D  P               INTEGER, SHORT, OR REAL?
811 0BB24 446           GOC    CPLX            IF NOT, GOTO CPLX
812 0BB27 A0F           D=D-1  P               REAL?
813 0BB2A 521           GONC   SHORT           IF NOT, GOTO SHORT
814 0BB2D F0            ASL    A
815 0BB2F 7D7E          GOSUB  ARYADR          COMPUTE REGISTER ADDRESS
816 0BB33 1567          C=DAT0 W               READ REGISTER
817 0BB37 8C00  ARYRTN  GOLONG =FNRTN3         PUSH VALUE & RETURN
          00
818
819 0BB3D D6    SHORT   C=A    A
820 0BB3F CA            A=A+C  A
821 0BB41 A0F           D=D-1  P               SHORT?
822 0BB44 540           GONC   INT             IF NOT, GOTO INT
823 0BB47 CA            A=A+C  A
824 0BB49 D6    INT     C=A    A
825 0BB4B CA            A=A+C  A
826 0BB4D 7D5E          GOSUB  2xRGNM          COMPUTE REGISTER ADDRESS
827 0BB51 181           D0=D0- 2
828 0BB54 17F           D1=D1+ 16
829 0BB57 8C00          GOLONG =INTSHT         PUSH VALUE & RETURN
          00
830
831 0BB5D 67B1  DOOFUS  GOTO   CNFLCT          Data Type error
832             *DOOFUS P=     0
833             *       LC(2)  =eDATTY
834             *       GOTO   nferr
835
836 0BB61 187   STRING  D0=D0- 8
837 0BB64 23            P=     3
838 0BB66 D2            C=0    A
839 0BB68 1561          C=DAT0 WP              READ MAXIMUM STRING LENGTH
840 0BB6C D7            D=C    A
841 0BB6E 167           D0=D0+ 8
842 0BB71 C6            C=C+C  A
843 0BB73 809           C+P+1
844 0BB76 8E00          GOSUBL =a-mult
          00
845 0BB7C 703E          GOSUB  ARYADR          COMPUTE REGISTER ADDRESS
```

```
846 0BB80 17F          D1=D1+ 16
847 0BB83 8C00         GOLONG =STRRCL        PUSH STRING & RETURN
          00
848
849 0BB89 D6    CPLX   C=A    A
850 0BB8B F0           ASL    A
851 0BB8D C6           C=C+C  A
852 0BB8F B07          D=D+1  P              12-DIGIT COMPLEX?
853 0BB92 440          GOC    CSHORT         IF NOT, GOTO CSHORT
854 0BB95 D6           C=A    A
855 0BB97 731E  CSHORT GOSUB  2xRGNM         COMPUTE REGISTER ADDRESS
856 0BB9B 132          ADOEX
857 0BB9E E7           D=D+1  A              SET TYPE INDICAT'N FOR CMPLX POLL
858 0BBA0 17F          D1=D1+ 16
859 0BBA3 817  =CSETUP DSRC
860 0BBA6 3176         LC(2)  =cRCL
861 0BBAA 6C70         GOTO   CPOLLR         GOTO CPOLLR
862
863             ************************************************************
864             ************************************************************
865             **
866             ** Name:    DMARRY  -  Evaluate Dummy Array Reference
867             **
868             ** Category:   FNEXEC
869             **
870             ** Purpose:
871             **      Evaluate dummy array reference.
872             **
873             ** Entry:
874             **      D0 = PC.
875             **      D1 = Mathstack pointer.
876             **      P=0.
877             **      Jumped on tDMYAR token.
878             **
879             ** Exit:
880             **      Through FNRTN2.
881             **
882             ** Calls:      VECCHK, RECADR.
883             **
884             ** History:
885             **
886             **    Date     Programmer             Modification
887             **  --------   ----------   ---------------------------------
888             **             SA           Wrote
889             **  10/13/83   NM           Attempted to document
890             **
891             ************************************************************
892             ************************************************************
893 0BBAE 8812         NIBHEX 8812
894 0BBB2 8E00 =DMARRY GOSUBL =ADDRSS
          00
895 0BBB8 101          R1=A
896 0BBBB 444          GOC    DMA100
897 0BBBE D1           B=0    A
898 0BBC0 AB5          B=C    X
```

```
 899 0BBC3 7A0E          GOSUB  VECCHK
 900 0BBC7 459           GOC    DOOFUS
 901 0BBCA 2E            P=     14
 902 0BBCC 320C          LCHEX  FC0
           F
 903 0BBD1 132           ADOEX
 904 0BBD4 D8            B=A    A
 905 0BBD6 AC5           B=C    S
 906 0BBD9 1560          C=DATO P
 907 0BBDD A85           B=C    P
 908 0BBE0 132           ADOEX
 909 0BBE3 20            P=     0
 910 0BBE5 903           ?C=D   P
 911 0BBE8 50            GOYES  DMA10
 912 0BBEA A45           B=B+B  S
 913 0BBED 1567  DMA10   C=DATO W
 914 0BBF1 8E00          GOSUBL =RECADR          RECTIFY VECTOR ADDRESS
           00
 915
 916 0BBF7 111   DMA20   A=R1
 917 0BBFA E4            A=A+1  A
 918 0BBFC 64C2          GOTO   fnrtn2
 919             *
 920 0BC00 75FD  DMA100  GOSUB  NEWVEC
 921 0BC04 7B2E          GOSUB  SAVSUB
 922 0BC08 AAB           C=D    XS
 923 0BC0B 31F9          LCHEX  9F
 924             ***************
 925             *       GONC   DMA20          BUGFIX 9605 ON 830803 -SA
 926             ***************
 927 0BC0F 47E           GOC    DMA20
 928             ***************
 929 0BC12 30C           LCHEX  C
 930             ***************
 931             *       GOC    DMA20          BUGFIX 9605 ON 830803 -SA
 932             ***************
 933 0BC15 51E           GONC   DMA20
 934             ********************************************************************
 935             ********************************************************************
 936             **
 937             ** Name:     CMPLX   -  Complex Poll
 938             ** Name:     CMPLXF  -  Complex Poll
 939             ** Name:     CMPLXR  -  Complex Poll
 940             ** Name:     CPOLL   -  Complex Poll
 941             **
 942             ** Category:   MTHUTL
 943             **
 944             ** Purpose:  Performs complex math poll.
 945             **
 946             ** Entry:    CMPLX is the Complex conversion function.
 947             **           CMPLXF assumes a function token lies left-
 948             **             justified in B(X). The token is shifted right.
 949             **             C is stored in R3. This is the normal entry
 950             **             point for most complex functions which override
 951             **             the mainframe functions. This entry point
```

```
952                **            assumes one of the POP routines was just called.
953                **          CMPLXR calls CPOLL, then jumps back into the
954                **            expression interpreter. This entry point is
955                **            used by recall operations. A(A) is assumed to
956                **            contain the variable address, with D(S) having
957                **            information about the variable format: even
958                **            for 12-digit, odd for 5-digit.
959                **
960                **          CPOLL does a fast poll to locate the complex
961                **            function Rom code. The function token is in
962                **            C(B). Storage functions have the same entry
963                **            conditions as for CMPLXR.
964                **
965                **
966                ** Exit:     CMPLX, CMPLXF, CMPLXR jump back to EXPR.
967                **          CPOLL returns with carry set, or else errors out.
968                **
969                ** Calls:    FPOLL
970                **
971                ** Uses:     A,B,C,D,R0,R1,R2,R3,R4,SCRATCH
972                **
973                ** Stk lvls: CPOLL: 3
974                **           CMPLX, CMPLXF, CPOLLF: 4
975                **
976                ** History:
977                **
978                **    Date     Programmer              Modification
979                ** --------   ----------   ----------------------------------
980                **             SA          Wrote
981                **  10/25/83   NM          Attempted to document
982                **
983                ********************************************************************
984                ********************************************************************
985 0BC18 8822           NIBHEX 8822
986 0BC1C 6800 =CMPLX    GOTO   CMPLX1
987
988 0BC20 108  =CPOLLF   R3=C
989 0BC23 F5             BSR    A
990 0BC25 D9    CMPLX1   C=B    A
991 0BC27 7B00 =CPOLLR   GOSUB  CPOLL          Entry point for rcl operations.
992 0BC2B 560            GONC   NOXLNK
993 0BC2E 68EC           GOTO   expr           Goto expr.
994 0BC32 6AE6  NOXLNK   GOTO   NOXFN
995
996 0BC36 101  =CPOLL    R1=A
997 0BC39 132            ADOEX
998 0BC3C 1B00           DO=(5) =FUNCDO
            000
999 0BC43 140            DATO=A A              Save PC in FUNCDO.
1000 0BC46 164           DO=DO+ 5
1001 0BC49 14C           DATO=C B              Save token in FUNCD1.
1002 0BC4C 1A00          DO=(4) =AVMEME
            00
1003 0BC52 137           CD1EX
1004 0BC55 144           DATO=C A              Save SP in AVMEME.
```

```
1005 0BC58 04          SETHEX
1006 0BC5A 8E00        GOSUBL =FPOLL          Perform poll..
          00
1007 0BC60 83          CON(2) =pCMPLX         ...for complex functions.
1008 0BC62 20          P=     0
1009 0BC64 1B00        D0=(5) =FUNCD0         Restore PC.
          000
1010 0BC6B 142         A=DAT0 A
1011 0BC6E 130         D0=A
1012 0BC71 831         ?XM=0                  Was poll handled?
1013 0BC74 00          RTNYES                 If so, return carry set
1014 0BC76 03          RTNCC                   else return carry clear.
1015            *      GOTO   NOXFN            else error out.
1016            *      GOTO   CNFLCT           else error out.
1017
1018 0BC78 8E00 =INVRES GOSUBL =INVNaN
          00
1019 0BC7E 8C00  ures12 GOLONG =uRES12
          00
1020            ********************************************************************
1021            ********************************************************************
1022            **
1023            ** Name:(S) OUTRES  -  Round And Return Result
1024            **
1025            ** Category:   EXCUTL
1026            **
1027            ** Purpose:
1028            **      Round result according to IEEE rounding rules, put on
1029            **      mathstack and reenter expression execution controller.
1030            **
1031            ** Entry:
1032            **      Result in (A,B), SB, XM and P as per uRES12 entry
1033            **        conditions.
1034            **      D1 = top of math stack.
1035            **
1036            ** Exit:
1037            **      Through EXPR.
1038            **
1039            ** Calls:      uRES12.
1040            **
1041            ** History:
1042            **
1043            **    Date     Programmer              Modification
1044            **  --------   ----------   ---------------------------------
1045            **             SA           Wrote
1046            **  11/01/83   NM           Attempted to document
1047            **
1048            ********************************************************************
1049            ********************************************************************
1050 0BC84 76FF =OUTRES GOSUB  ures12
1051 0BC88 6751         GOTO   FN4
1052            ********************************************************************
1053            ********************************************************************
1054            **
1055            ** Name:(S) pCMPLX  -  Complex Number Operation Poll
```

```
1056            **
1057            ** Category:   POLL
1058            **
1059            ** Type:       FPOLL
1060            **
1061            ** Purpose:
1062            **      Look for handler to perform complex operation:
1063            **      Function, Store or Recall.
1064            **
1065            ** Should poll be "Handled" (return with XM=0)?:
1066            **      Yes.
1067            **
1068            ** Meaning of "Handling" Poll (what does code do if handled?):
1069            **      Handler has performed complex operation.  If poll is
1070            **      not handled, calling code errors out (eDATTY).
1071            **
1072            ** Entry conditions for handler (registers, ST, RAM, etc.):
1073            **      Carry set on entry.
1074            **      B[A] = Poll number.
1075            **      HEX mode.
1076            **      P=0.
1077            **      (FUNCD0) contains PC, pointing past token.
1078            **      (FUNCD1) contains 2-nibble token.
1079            **      (AVMEME) contains stack pointer.
1080            **      If token is a function token of one parameter (e.g.,
1081            **        SIN(Z)), then R1 = Real part of argument,
1082            **                      R0 = Imaginary part of argument.
1083            **      If token is a function token of two parameters (e.g.,
1084            **        Z*W), then R0 = Imaginary part of argument at top
1085            **                        of stack (second argument),
1086            **                    R1 = Real part of second argument.
1087            **                    R2 = Imaginary part of first argument.
1088            **                    R3 = Real part of first argument.
1089            **        If either argument is real, the imaginary part will
1090            **        be represented as 0000000000000900.
1091            **      If token is a comparison token, entry conditions are
1092            **        the same as for other two-parameter functions.  The
1093            **        predicate can be obtained by looking at (PC). (or
1094            **        maybe (PC-1)?)
1095            **      If token is cR->C, it means that a real value is being
1096            **        assigned to a variable whose type is not real, short
1097            **        or integer.  The value to be assigned is at the top
1098            **        of the stack and the variable destination information
1099            **        occupies STMT scratch as set up by DEST routine.
1100            **      If token is cC->C, it means that a value which is
1101            **        neither real or string is being assigned to some
1102            **        variable.  The value to be assigned is at the top of
1103            **        the stack and the variable destination information
1104            **        occupies STMT scratch as set up by DEST routine.
1105            **      If token is cRCL, it means that a complex number needs
1106            **        to be recalled (put on the stack).
1107            **          R1[A] points at the value to be recalled.
1108            **          D[S] is odd iff value is COMPLEX SHORT.
1109            **
1110            ** Normal exit conditions from handler if handled (ST, RAM,
```

```
1111            ** registers, etc.):
1112            **      HEX mode.
1113            **      XM=0.
1114            **      For functions and comparisons, result pushed on
1115            **        math stack (handler must do available memory check
1116            **        and error out if insufficient memory), complete with
1117            **        stack signature.  D1 = stack pointer.
1118            **      For store, no further exit conditions.
1119            **      For recall (token = cRCL), value has been pushed on
1120            **        stack, D1 = stack pointer, B[A] = address of
1121            **        variable register, B[S] = E iff COMPLEX, F iff SHORT
1122            **        COMPLEX.
1123            **
1124            ** Normal exit conditions from handler if not handled (ST, RAM,
1125            ** registers, etc.):
1126            **      HEX mode.
1127            **      XM=1.
1128            **
1129            ** Available subroutine levels:
1130            **      1
1131            **
1132            ** What registers/RAM may be used if handled?:
1133            **      A-D,D0,D1,P,R0-R4, function scratch RAM.
1134            **
1135            ** What registers/RAM may be used if not handled?:
1136            **      A-C, D[15-5] D0, D1, P.
1137            **
1138            ** Envisioned application(s):
1139            **      Extension of mainframe functions to complex arguments.
1140            **
1141            ** History:
1142            **
1143            **    Date     Programmer             Modification
1144            **  --------   ----------   ----------------------------------
1145            **             SA           Wrote
1146            **  10/20/83   NM           Attempted to document
1147            **
1148            ************************************************************************
1149            ************************************************************************
1150            *
1151            ************************************************************************
1152            ************************************************************************
1153            **
1154            ** Name:(S) POP2N   -  Pop 2 Numbers From Stack.
1155            **
1156            ** Category:   MTHSTK
1157            **
1158            ** Purpose:
1159            **      Pop 2 numbers from math stack.
1160            **
1161            ** Entry:
1162            **      D1=Stack pointer.
1163            **
1164            ** Exit:
1165            **      DEC mode.
```

```
1166            **      D1 16 nibbles before end of entry (D1=D1+16 to get to
1167            **        next entry.
1168            **      If carry clear:
1169            **        C[W] = first number on stack.
1170            **        A[W] = second number on stack.
1171            **      If carry set {one or both numbers complex}:
1172            **        C[W]=Real part of first number.
1173            **        R2=Imaginary part of first number.
1174            **        A[W]=Real part of second number.
1175            **        R0=Imaginary part of second number.
1176            **        Imaginary part = 0000000000000900 if arg is real.
1177            **      Error exit (eDATTY) if either arg not numeric.
1178            **
1179            ** Calls:      None.
1180            **
1181            ** Uses.......
1182            **             A,B[0],C,P.  If Carry Set: R0, R2.
1183            **
1184            ** Stk lvls:   0
1185            **
1186            ** History:
1187            **
1188            **    Date      Programmer              Modification
1189            **  --------    ----------   ----------------------------------
1190            **              SA           Wrote
1191            **  10/13/83    NM           Attempted to document
1192            **
1193            ********************************************************************
1194            ********************************************************************
1195 0BC8C 05   =POP2N  SETDEC
1196 0BC8E 20           P=      0
1197 0BC90 BB1          BSL     X
1198 0BC93 A0D          B=B-1   P               CREATE TEST DIGIT
1199 0BC96 1577         C=DAT1  W               READ STACK REGISTER
1200 0BC9A 985          ?C>B    P               REAL?
1201 0BC9D 01           GOYES   RSLV10          IF NOT, GOTO RSLV10
1202 0BC9F 17F          D1=D1+  16
1203 0BCA2 1537         A=DAT1  W               READ STACK REGISTER
1204 0BCA6 980          ?A>B    P               REAL?
1205 0BCA9 E3           GOYES   RSLV20          IF NOT, GOTO RSLV20
1206 0BCAB 03           RTNCC                   RETURN CARRY CLEAR
1207
1208 0BCAD 04    RSLV10 SETHEX
1209 0BCAF B06          C=C+1   P
1210 0BCB2 B06          C=C+1   P
1211 0BCB5 96E          ?C#0    B               COMPLEX?
1212 0BCB8 D5           GOYES   CNFLCT          IF NOT, GOTO CNFLCT
1213 0BCBA 171          D1=D1+  2
1214 0BCBD 1577         C=DAT1  W               READ IMAGINARY PART
1215 0BCC1 17F          D1=D1+  16
1216 0BCC4 10A          R2=C
1217 0BCC7 1577         C=DAT1  W               READ REAL PART
1218 0BCCB 17F          D1=D1+  16
1219 0BCCE 1537         A=DAT1  W               READ STACK REGISTER
1220 0BCD2 980          ?A>B    P               REAL?
```

```
1221 0BCD5 E1          GOYES  RSLV30        IF NOT, GOTO RSLV30
1222 0BCD7 100         RO=A
1223 0BCDA AF0         A=0    W             CONVERT TO COMPLEX
1224 0BCDD 05          SETDEC
1225 0BCDF A2C         A=A-1  XS            A[M]=0 & A[XS]=9 ==> REAL
1226 0BCE2 120         AROEX
1227 0BCE5 02          RTNSC                RETURN CARRY SET
1228 0BCE7 10A  RSLV20 R2=C
1229 0BCEA AF2         C=0    W             CONVERT TO COMPLEX
1230 0BCED A2E         C=C-1  XS            (indicates REAL arg)
1231 0BCF0 12A         CR2EX
1232 0BCF3 04   RSLV30 SETHEX
1233 0BCF5 B04         A=A+1  P
1234 0BCF8 B04         A=A+1  P
1235 0BCFB 96C         ?A#0   B             COMPLEX?
1236 0BCFE 71          GOYES  CNFLCT        IF NOT, GOTO CNFLCT
1237 0BD00 171         D1=D1+ 2
1238 0BD03 1537        A=DAT1 W             READ IMAGINARY PART
1239 0BD07 17F         D1=D1+ 16
1240 0BD0A 100         RO=A
1241 0BD0D 1537        A=DAT1 W             READ REAL PART
1242 0BD11 05          SETDEC
1243 0BD13 02          RTNSC                RETURN CARRY SET
1244
1245          ************************************************************
1246          ************************************************************
1247          **
1248          ** Name:(S) CNFLCT  -  Report "Data Type" Error.
1249          **
1250          ** Category:   SYSTEM
1251          **
1252          ** Purpose:
1253          **      To do a GOVLNG =RDATTY.
1254          **
1255          ** History:
1256          **
1257          **    Date     Programmer          Modification
1258          **  --------   ----------   ----------------------------------
1259          **  11/09/83   MB           Documentation
1260          **
1261          ************************************************************
1262          ************************************************************
1263 0BD15 8D00 =CNFLCT GOVLNG =RDATTY
         000
1264          ************************************************************
1265          ************************************************************
1266          **
1267          ** Name:(S) POP1N   -  Pop 1 Number Off Of Stack
1268          **
1269          ** Category:   MTHSTK
1270          **
1271          ** Purpose:
1272          **      Pop one numeric value off of math stack.
1273          **
1274          ** Entry:
```

```
1275            **      D1 = Stack pointer.
1276            **
1277            ** Exit:
1278            **      Errors out (eDATTY) if non-numeric item.
1279            **      DEC mode.
1280            **      P=0.
1281            **      If carry clear: Result real.
1282            **        Result in A.
1283            **      If carry set: Result complex.
1284            **        Real part in A.
1285            **        Imaginary part in R0.
1286            **
1287            ** Calls:       None.
1288            **
1289            ** Uses.......
1290            **              A,B[0].  If carry set, R0.
1291            **
1292            ** Stk lvls:    0
1293            **
1294            ** History:
1295            **
1296            **    Date      Programmer              Modification
1297            **  --------    ----------    ----------------------------------
1298            **              SA            Wrote
1299            **  10/13/83    NM            Attempted to document
1300            **
1301            ********************************************************************
1302            ********************************************************************
1303 0BD1C 05   =POP1N  SETDEC
1304 0BD1E 20          P=      0
1305 0BD20 BB1         BSL     X
1306 0BD23 A0D         B=B-1   P               CREATE TEST DIGIT
1307 0BD26 1537        A=DAT1  W               READ STACK REGISTER
1308 0BD2A 980         ?A>B    P               REAL?
1309 0BD2D 6C          GOYES   RSLV30          IF NOT, GOTO RSLV30
1310 0BD2F 03          RTNCC                   RETURN CARRY CLEAR
1311            ********************************************************************
1312            ********************************************************************
1313            **
1314            ** Name:(S) REVPOP  -  REV$ On String And Then POP1S
1315            **
1316            ** Category:    MTHSTK
1317            **
1318            ** Purpose:
1319            **      Reverse a string on the stack and then pop it.
1320            **
1321            ** Entry:
1322            **      D1=Mathstack pointer.
1323            **      HEX mode.
1324            **
1325            ** Exit:
1326            **      A[A]=string length.
1327            **      D1 pointing at low-address end of string (last char).
1328            **      P=0.
1329            **
```

```
1330            ** Calls:      REV$, POP1S (falls through).
1331            **
1332            ** Uses.......
1333            **             A,B,C[A],D[A],P,D1
1334            **
1335            ** Stk lvls:   2
1336            **
1337            ** History:
1338            **
1339            **    Date     Programmer             Modification
1340            **  --------   ----------   ----------------------------------
1341            **             SA           Wrote
1342            **  10/13/83   NM           Attempted to document
1343            **
1344            ************************************************************************
1345            ************************************************************************
1346 0BD31 8F00 =REVPOP GOSBVL =REV$
           000
1347            ************************************************************************
1348            ************************************************************************
1349            **
1350            ** Name:(S) POP1S   -  Pop 1 String Arg Off Stack
1351            **
1352            ** Category:   MTHSTK
1353            **
1354            ** Purpose:
1355            **      Position pointers to pop a string argument off of
1356            **      math stack.
1357            **
1358            ** Entry:
1359            **      HEX mode.
1360            **      D1 pointing at string header in stack.
1361            **
1362            ** Exit:
1363            **      Errors out (Data type) if item on stack is not string.
1364            **      P=0.
1365            **      D1 pointing past string header... pointing at last
1366            **        character of string.
1367            **      A[A]=length of string in nibbles.
1368            **
1369            ** Calls:      None.
1370            **
1371            ** Uses.......
1372            **             A[W],D1,P
1373            **
1374            ** Stk lvls:   0
1375            **
1376            ** NOTE:
1377            **      Does not return if item on stack is not string.
1378            **
1379            ** History:
1380            **
1381            **    Date     Programmer             Modification
1382            **  --------   ----------   ----------------------------------
1383            **             SA           Wrote
```

```
1384            **  09/23/83   NM          Attempted to document
1385            **
1386            ********************************************************************
1387            ********************************************************************
1388 0BD38 15B6 =POP1S  A=DAT1 7             READ TYPE INDICATOR
1389 0BD3C 20    POP1S1 P=     0
1390 0BD3E B04         A=A+1  P
1391 0BD41 A64         A=A+A  B
1392 0BD44 96C         ?A#0   B             ACTUAL STRING?
1393 0BD47 EC          GOYES  CNFLCT        IF NOT, GOTO POPERR
1394 0BD49 BF4   POP1S2 ASR    W
1395 0BD4C BF4         ASR    W
1396 0BD4F 17F         D1=D1+ 16
1397 0BD52 03          RTNCC                RETURN
1398            ********************************************************************
1399            ********************************************************************
1400            **
1401            ** Name:(S) MPOP2N  -  Pop 2 Args W/signan Check
1402            ** Name:(S) POP2N+  -  Pop 2 Args W/signan Check
1403            **
1404            ** Category:   MTHSTK
1405            **
1406            ** Purpose:
1407            **      Pop two arguments off of the math stack and report
1408            **      signaling NANs.
1409            **      MPOP2N calls uMODES to fetch modes to ST.
1410            **      POP2N+ assumes this has already been done.
1411            **
1412            ** Entry:
1413            **      D1 = stack pointer.
1414            **
1415            ** Exit:
1416            **      Carry set: One or both numbers are complex.  signaling
1417            **        NaN check not done.  Same exit conditions as POP2N.
1418            **      Carry clear: C[W] = first number on stack.
1419            **                   A[W] = second number on stack.
1420            **      P=0.
1421            **      D1 pointing 16 nibbles before next stack entry.
1422            **
1423            ** Calls:      POP2N, SIGTST, URES12, uMODES.
1424            **
1425            ** Uses.......
1426            **             A,B,C,D,R3,S7-S11.
1427            **
1428            ** Stk lvls:   3
1429            **
1430            ** History:
1431            **
1432            **    Date     Programmer              Modification
1433            **  --------   ----------   ------------------------------------
1434            **             SA           Wrote
1435            **  10/14/83   NM           Attempted to document
1436            **
1437            ********************************************************************
1438            ********************************************************************
```

```
1439 0BD54 7950 =MPOP2N GOSUB  uMODES
1440 0BD58 703F =POP2N+ GOSUB  POP2N
1441 0BD5C 400          RTNC
1442 0BD5F 108          R0=C
1443 0BD62 8E00         GOSUBL =SIGTST        check for signaling NaN
          00
1444 0BD68 590          GONC   POP2NA         jump if not a signaling NaN
1445 0BD6B 7F0F         GOSUB  ures12         process signaling NaN
1446 0BD6F AFA          A=C    W
1447 0BD72 120   POP2NA AROEX
1448 0BD75 8E00         GOSUBL =SIGTST        check for signaling NaN
          00
1449 0BD7B 590          GONC   POP2NB         jump if not a signaling NaN
1450 0BD7E 7CFE         GOSUB  ures12         process signaling NaN
1451 0BD82 AFA          A=C    W
1452 0BD85 AF6   POP2NB C=A    W
1453 0BD88 110          A=R0
1454 0BD8B 03           RTNCC
1455             ********************************************************************
1456             ********************************************************************
1457             **
1458             ** Name:(S) MPOP1N  -  Pop 1 Arg & Check For Sig NaN
1459             ** Name:(S) POP1N+  -  Pop 1 Arg & Check For Sig NaN
1460             **
1461             ** Category:   MTHSTK
1462             **
1463             ** Purpose:
1464             **      Pop one numeric argument and give Signaled Op message
1465             **      if appropriate.
1466             **
1467             ** Entry:
1468             **      D1=Mathstack pointer
1469             **   POP1N+: S8-S11 already set according to modes (uMODES)
1470             **           has already been called.
1471             **
1472             ** Exit:
1473             **      DEC mode.
1474             **      Carry set: Result is complex.  Signaling NaN check not
1475             **                 performed.  Result in A/R0 as per POP1N.
1476             **
1477             ** Calls:      uMODES, POP1N, SIGTST, uRES12.
1478             **
1479             ** Uses.......
1480             **             A,B,C,D,R3,S8-S11.
1481             **
1482             ** Stk lvls:   3
1483             **
1484             ** History:
1485             **
1486             **    Date     Programmer              Modification
1487             **  --------   ----------   ----------------------------------
1488             **             SA           Wrote
1489             **  10/14/83   NM           Attempted to document
1490             **
1491             ********************************************************************
```

```
1492            ********************************************************************
1493            *
1494            ********************************************************************
1495            ********************************************************************
1496            **
1497            ** Name:(S) SIGCHK  -  Report Signaling NaN
1498            **
1499            ** Category:   MTHUTL
1500            **
1501            ** Purpose:
1502            **      Check for signaling NaN and report "Signaled Op" if
1503            **      found.
1504            **
1505            ** Entry:
1506            **      Number in A.
1507            **      DEC mode.
1508            **
1509            ** Exit:
1510            **      Number in A.
1511            **      Carry clear.
1512            **
1513            ** Calls:      uRES12, SIGTST.
1514            **
1515            ** Uses.......
1516            **              A-D,P,R3,S7-S11.
1517            **
1518            ** Stk lvls:   3
1519            **
1520            ** History:
1521            **
1522            **    Date      Programmer              Modification
1523            **  --------    ----------   ----------------------------------
1524            **              SA           Wrote
1525            **  11/01/83    NM           Attempted to document
1526            **
1527            ********************************************************************
1528            ********************************************************************
1529 0BD8D 7020 =MPOP1N GOSUB  uMODES
1530 0BD91 778F =POP1N+ GOSUB  POP1N
1531 0BD95 400          RTNC
1532 0BD98 8E00 =SIGCHK GOSUBL =SIGTST        Check for signaling NaN.
          00
1533 0BD9E 500          RTNNC                 NOT a Signaling NaN .
1534 0BDA1 79DE         GOSUB  ures12         process signaling NaN
1535 0BDA5 AFA          A=C    W
1536 0BDA8 03           RTNCC
1537            ********************************************************************
1538 0BDAA 8D00 =fltdh  GOVLNG =FLTDH
          000
1539            ********************************************************************
1540            ********************************************************************
1541            **
1542            ** Name:    uMODES  -  Set S8-S11 To Indicate Modes
1543            **
1544            ** Category:   MTHUTL
```

```
1545                ** 
1546                ** Purpose:
1547                **      Set S8-S11 according to math modes.
1548                ** 
1549                ** Entry:
1550                **      Yes.
1551                ** 
1552                ** Exit:
1553                **              S8 - User mode if set
1554                **              S9 - Rad Mode if set
1555                **              S10- sINFRD (Rounding mode)
1556                **              S11- sNEGRD (Rounding mode)
1557                ** 
1558                **              S10  S11    ROUND
1559                **              ---  ---  -----------
1560                **               0    0   Nearest
1561                **               0    1   Toward zero
1562                **               1    0   Toward +INF
1563                **               1    1   Toward -INF
1564                ** 
1565                ** Calls:       None.
1566                ** 
1567                ** Uses.......
1568                **              D[A], S8-S11.
1569                ** 
1570                ** Stk lvls:    0
1571                ** 
1572                ** History:
1573                ** 
1574                **    Date      Programmer              Modification
1575                **  --------    ----------   ----------------------------------
1576                **              SA           Wrote
1577                **  10/14/83    NM           Attempted to document
1578                ** 
1579                ********************************************************************
1580                ********************************************************************
1581 0BDB1 D7     =uMODES D=C     A
1582 0BDB3 137            CD1EX                   Save D1
1583 0BDB6 1F00           D1=(5) (=FLGREG)-#E     RND,INF,RAD Mode address
           000
1584 0BDBD 0B             CSTEX                   Fetch current Status
1585 0BDBF 1572           C=DAT1 XS               Set S8-S11 as above
1586 0BDC3 0B             CSTEX                   Restore Status
1587 0BDC5 135            D1=C                    Restore D1
1588 0BDC8 DB             C=D     A
1589 0BDCA 03             RTNCC
1590                ********************************************************************
1591                ********************************************************************
1592                ** 
1593                ** Name:     ASIN     -  ASIN Function
1594                ** Name:     ACOS     -  ACOS Function
1595                ** Name:     ATAN     -  ATAN Function
1596                ** Name:     ANGLE    -  ANGLE Function
1597                ** Name:     CHR$     -  CHR$ Function
1598                ** Name:     SIN      -  SIN Function
```

```
1599                ** Name:    COS     -  COS Function
1600                ** Name:    TAN     -  TAN Function
1601                **
1602                ** Category:   FNEXEC
1603                **
1604                ** Purpose:
1605                **      Assorted mainframe functions.
1606                **
1607                ** Entry:
1608                **      D0 = PC.
1609                **      D1 = Mathstack pointer.
1610                **      P=0.
1611                **      Jumped on function token.
1612                **
1613                ** Exit:
1614                **      Through EXPR.
1615                **
1616                ** Calls:      Whatever.
1617                **
1618                ** Detail:
1619                **      SIN, COS, TAN, ASIN, ACOS, ATAN are numeric functions
1620                **        of one numeric argument.
1621                **      ANGLE is a numeric function of two numeric arguments.
1622                **      CHR$ is a string function of one numeric argument.
1623                **
1624                ** History:
1625                **
1626                **    Date     Programmer              Modification
1627                **  --------   ----------   ------------------------------------
1628                **             SA           Wrote
1629                **  10/14/83   NM           Attempted to document
1630                **
1631                ************************************************************
1632                ************************************************************
1633 0BDCC 811           NIBHEX 811            Argument count range = [1,1]
1634 0BDCF 7ABF =ASIN    GOSUB  MPOP1N         Trig mode lookup
1635 0BDD3 487           GOC    CLNK4
1636 0BDD6 8E00          GOSUBL =ASIN12
          00
1637 0BDDC 67AE          GOTO   OUTRES
1638                ************************************************************
1639 0BDE0 8C00  FN4     GOLONG =FNRTN4
          00
1640                ************************************************************
1641 0BDE6 811           NIBHEX 811            Argument count range = [1,1]
1642 0BDE9 70AF =ACOS    GOSUB  MPOP1N         Trig mode lookup
1643 0BDED 4E5           GOC    CLNK4
1644 0BDF0 8E00          GOSUBL =ACOS12
          00
1645 0BDF6 6D8E          GOTO   OUTRES
1646                ************************************************************
1647 0BDFA 811           NIBHEX 811            Argument count range = [1,1]
1648 0BDFD 7C8F =ATAN    GOSUB  MPOP1N         Trig mode lookup
1649 0BE01 4A4           GOC    CLNK4
1650 0BE04 8E00          GOSUBL =ATAN12
```

```
           00
1651 0BE0A 697E          GOTO   OUTRES
1652                     ************************************************************
1653 0BE0E 8822          NIBHEX 8822            Argument count range = [1,1]
1654 0BE12 7E3F =ANGLE   GOSUB  MPOP2N          Trig mode lookup
1655 0BE16 453           GOC    CLNK4
1656 0BE19 8E00          GOSUBL =ARG12
           00
1657 0BE1F 646E          GOTO   OUTRES
1658                     ************************************************************
1659 0BE23 811           NIBHEX 811             Argument count range = [1,1]
1660 0BE26 736F =CHR$    GOSUB  MPOP1N          POP ARGUMENT
1661 0BE2A 7C7F          GOSUB  fltdh           CONVERT ARGUMENT TO HEX
1662 0BE2E AF2           C=0    W
1663 0BE31 831           ?XM=0                  REALISTIC ARGUMENT?
1664 0BE34 90            GOYES  CHR$1           IF SO, GOTO CHR$1
1665 0BE36 30F           LCHEX  F
1666 0BE39 66AF          GOTO   FN4             WRITE NULL STRING TO STACK
1667 0BE3D 17D  CHR$1    D1=D1+ 14
1668 0BE40 149           DAT1=A B               WRITE CHARACTER TO STACK
1669 0BE43 32F0          LCHEX  20F
           2
1670 0BE48 6079  fn1     GOTO   FN1             WRITE STRING HEADER & RETURN
1671                     ************************************************************
1672 0BE4C 63DD  CLNK4   GOTO   CPOLLF
1673
1674 0BE50 811           NIBHEX 811             Argument count range = [1,1]
1675 0BE53 763F =SIN     GOSUB  MPOP1N          Trig mode lookup
1676 0BE57 44F           GOC    CLNK4
1677 0BE5A 8E00          GOSUBL =SIN12
           00
1678 0BE60 632E          GOTO   OUTRES
1679                     ************************************************************
1680 0BE64 811           NIBHEX 811             Argument count range = [1,1]
1681 0BE67 722F =COS     GOSUB  MPOP1N          Trig mode lookup
1682 0BE6B 40E           GOC    CLNK4
1683 0BE6E 8E00          GOSUBL =COS12
           00
1684 0BE74 6F0E          GOTO   OUTRES
1685                     ************************************************************
1686 0BE78 811           NIBHEX 811             Argument count range = [1,1]
1687 0BE7B 7E0F =TAN     GOSUB  MPOP1N          Trig mode lookup
1688 0BE7F 4CC           GOC    CLNK4
1689 0BE82 8E00          GOSUBL =TAN12
           00
1690 0BE88 6BFD          GOTO   OUTRES
1691                     ************************************************************
1692                     ****   COT,CSC, & SEC removed from mainframe - 3/31/83   S.B.
1693                     ****
1694                     ************************************************************
1695
1696                     ************************************************************
1697                     ************************************************************
1698                     **
1699                     ** Name:    ERRN    -  ERRN Function
```

```
1700                    ** Name:    ERRL    -  ERRL Function
1701                    **
1702                    ** Category:   FNEXEC
1703                    **
1704                    ** Purpose:
1705                    **      Evaluate ERRN, ERRL functions.
1706                    **
1707                    ** Entry:
1708                    **      D0 = PC.
1709                    **      D1 = Mathstack pointer.
1710                    **      P=0.
1711                    **      Jumped on function token.
1712                    **
1713                    ** Exit:
1714                    **      Through EXPR.
1715                    **
1716                    ** Calls:      Whatever.
1717                    **
1718                    ** Detail:
1719                    **      These are both no-argument functions.
1720                    **
1721                    ** History:
1722                    **
1723                    **    Date      Programmer              Modification
1724                    **  --------    ----------   ------------------------------------
1725                    **              SA           Wrote
1726                    **  10/25/83    NM           Attempted to document
1727                    **
1728                    ************************************************************
1729                    ************************************************************
1730 0BE8C 00                 NIBHEX 00              Argument count range = [0,0]
1731                    *
1732                    *  Changed by MB 1/8/82
1733                    *   ERR# is stored as 4 nib hex number: (LEX ID#)*256 + msg# in
1734                    *   hex.
1735                    *   This is converted to  (LEX ID#)*1000 + msg#  in decimal.
1736                    *     e.g.  ERR#= 171B  is converted to  23027
1737                    *
1738 0BE8E 3400 =ERRN    LC(5)  (=ERR#)+2       Point to LEX ID#.
          000
1739 0BE95 136           CDOEX
1740 0BE98 D7            D=C    A               Save D0.
1741 0BE9A AF0           A=0    W
1742 0BE9D AF2           C=0    W
1743 0BEA0 14A           A=DAT0 B               Read in LEX ID#.
1744 0BEA3 328E          LC(3)  1000
          3
1745 0BEA8 8E00          GOSUBL =a-mult         Multiply LEX ID# by 1000.
          00
1746 0BEAE D2            C=0    A               Clr C(A) since A-MULT don't.(SB)
1747 0BEB0 181           D0=D0- 2
1748 0BEB3 14E           C=DAT0 B               Add message number.
1749 0BEB6 CA            A=A+C  A
1750 0BEB8 7B00  HDFRT2  GOSUB  hdflt           Convert all to dec float.
1751 0BEBC DB            C=D    A               C= previous D0.
```

```
1752 0BEBE AFE   ACfrt2 ACEX   W
1753 0BEC1 8C00  fnrtn2 GOLONG =FNRTN2
           00
1754 0BEC7 8D00 =hdflt  GOVLNG =HDFLT
           000
1755            ****************************************************************
1756 0BECE 00           NIBHEX 00             Argument count range = [0,0]
1757 0BED0 3400 =ERRL   LC(5)  =ERRL#
           000
1758 0BED7 136          CDOEX
1759 0BEDA AF0          A=0    W
1760 0BEDD 15A3         A=DAT0 4
1761 0BEE1 8E00         GOSUBL =FLOATA
           00
1762 0BEE7 66DF         GOTO   ACfrt2
1763            ****************************************************************
1764            ****************************************************************
1765            **
1766            ** Name:     EXP     -  EXP Function
1767            ** Name:     EXPM1   -  EXPM1 Function
1768            ** Name:     LEN     -  LEN Function
1769            ** Name:     LOG     -  LOG Function
1770            ** Name:     LOG10   -  LOG10 Function
1771            ** Name:     LOGP1   -  LOGP1 Function
1772            ** Name:     MEM     -  MEM Function
1773            ** Name:     PI      -  PI Function
1774            ** Name:     POS     -  POS Function
1775            ** Name:     RMD     -  RMD Function
1776            ** Name:     MOD     -  MOD Function
1777            ** Name:     RED     -  RED Function
1778            ** Name:     DIV     -  DIV Function
1779            ** Name:     DEG     -  DEG Function
1780            ** Name:     SQR     -  SQR Function
1781            ** Name:     RAD     -  RAD Function
1782            **
1783            ** Category:   FNEXEC
1784            **
1785            ** Purpose:
1786            **      Function execute code for various functions.
1787            **
1788            ** Entry:
1789            **      P=0.
1790            **      D0 = PC.
1791            **      D1 = Mathstack pointer.
1792            **      Jumped on function tokens.
1793            **
1794            ** Exit:
1795            **      Through EXPR.
1796            **
1797            ** Calls:      Whatever.
1798            **
1799            ** Detail:
1800            **      Numeric functions of no arguments:
1801            **        MEM (if port# not specified),
1802            **        PI (pi).
```

```
1803            **      Numeric functions of one numeric argument:
1804            **        EXP (E^X),
1805            **        EXPM1 (E^X-1),
1806            **        LOG (LN),
1807            **        LOG10 (LOG base 10),
1808            **        LOGP1 (LN(x) + 1),
1809            **        MEM (if port# specified),
1810            **        DEG (radians to degrees),
1811            **        SQR (square root),
1812            **        RAD (degrees to radians).
1813            **      Numeric functions of two numeric arguments:
1814            **        RMD (remainder),
1815            **        MOD (mod),
1816            **        RED (reduction),
1817            **        DIV (DIV operator)
1818            **      Numeric functions of one string argument:
1819            **        LEN (length of string).
1820            **      Numeric functions of two string arguments:
1821            **        POS (position function).
1822            **
1823            ** History:
1824            **
1825            **    Date     Programmer             Modification
1826            **  --------   ----------   ----------------------------------
1827            **               SA          Wrote
1828            **  10/14/83     NM          Attempted to document
1829            **
1830            ****************************************************************
1831            ****************************************************************
1832 OBEEB 811          NIBHEX 811            Argument count range = [1,1]
1833 OBEEE 7B9E =EXP    GOSUB  MPOP1N         POP ARGUMENT
1834 OBEF2 486          GOC    CLNK5
1835 OBEF5 8E00         GOSUBL =EXP12
          00
1836 OBEFB 688D         GOTO   OUTRES
1837            ****************************************************************
1838 OBEFF 811          NIBHEX 811            Argument count range = [1,1]
1839 OBF02 778E =EXPM1  GOSUB  MPOP1N
1840 OBF06 445          GOC    CLNK5
1841 OBF09 8E00         GOSUBL =EX-112
          00
1842 OBF0F 8E00         GOSUBL =EXAB1         get [exp(x)-1] from (R0,R1) regs
          00
1843 OBF15 6E6D         GOTO   OUTRES
1844            *
```

```
1845                    EJECT
1846                ****************************************************************
1847                ****************************************************************
1848                **
1849                ** Name:(S) ARGERR  -  Report "Invalid Arg" Error.
1850                **
1851                ** Category:   SYSTEM
1852                **
1853                ** Purpose:
1854                **      To report "Invalid Arg" as an execution error.
1855                **
1856                ** Entry:
1857                **      S13=0 if not a running program (i.e., keyboard
1858                **            execution error)
1859                **      S13=1 if a running program.
1860                **      No other necessary conditions.
1861                **
1862                ** Exit:
1863                **      Exits to BASIC main loop (ERRRTN)
1864                **
1865                ** Calls:      MFERR
1866                **
1867                ** Uses....... Exits to main loop, can use anything
1868                **
1869                ** Stk lvls:   Exits to main loop, can use all
1870                **
1871                ** NOTE:
1872                **      ARGERR sets P=0 to select an execution error:
1873                **        -- not a parse error
1874                **        -- store ERRN (and ERRL, if S13=1)
1875                **        -- display "ERR:" (or "ERR L<#>:") prefix
1876                **        -- exit to BASIC main loop
1877                **
1878                ** Detail:
1879                **     =ARGERR P=     0
1880                **             LC(2)  =eIVARG
1881                **             GOLONG =MFERR
1882                **
1883                ** History:
1884                **
1885                **    Date     Programmer          Modification
1886                **  --------   ----------   ----------------------------------
1887                **  11/09/83   MB           Documentation
1888                **
1889                ****************************************************************
1890                ****************************************************************
1891 0BF19 20      =ARGERR P=     0
1892 0BF1B 3100         LC(2)  =eIVARG        Invalid Argument
1893 0BF1F 8C00  mferr  GOLONG =MFERR         EXECUTION ERROR!!!
                 00
1894                ****************************************************************
1895 0BF25 411          NIBHEX 411            Argument count range = [1,1]
1896 0BF28 7C0E =LEN    GOSUB  POP1S          READ HEADER
1897 0BF2C 137          CD1EX
1898 0BF2F C2           C=C+A  A              COMPUTE ADDRESS OF RESULT
```

```
1899 0BF31 135          D1=C
1900 0BF34 1CF          D1=D1- 16
1901 0BF37 25    LEN10  P=     5              CONVERT LENGTH TO BYTES
1902 0BF39 A80          A=0    P
1903 0BF3C 81C          ASRB
1904 0BF3F 748F =LEN20  GOSUB  hdflt          CONVERT HEX TO DECIMAL
1905 0BF43 6FC9         GOTO   D1AEXP         WRITE RESULT TO STACK, RETURN
1906                    ****************************************************************
1907 0BF47 811          NIBHEX 811            Argument count range = [1,1]
1908 0BF4A 7F3E =LOG    GOSUB  MPOP1N         POP ARGUMENT
1909 0BF4E 4C0          GOC    CLNK5
1910 0BF51 8E00         GOSUBL =LN12
             00
1911 0BF57 6C2D         GOTO   OUTRES
1912                    ****************************************************************
1913 0BF5B 64CC  CLNK5  GOTO   CPOLLF
1914
1915 0BF5F 811          NIBHEX 811            Argument count range = [1,1]
1916 0BF62 772E =LOG10  GOSUB  MPOP1N         POP ARGUMENT
1917 0BF66 44F          GOC    CLNK5
1918 0BF69 8E00         GOSUBL =LGT12
             00
1919 0BF6F 641D         GOTO   OUTRES
1920                    ****************************************************************
1921 0BF73 811          NIBHEX 811            Argument count range = [1,1]
1922 0BF76 731E =LOGP1  GOSUB  MPOP1N
1923 0BF7A 40E          GOC    CLNK5
1924 0BF7D 8E00         GOSUBL =LN1+12
             00
1925 0BF83 600D         GOTO   OUTRES
1926                    ****************************************************************
1927 0BF87 3100  MEMeDV LC(2)  =eDVCNF
1928 0BF8B 639F         GOTO   mferr
1929
1930 0BF8F 801          NIBHEX 801            Argument count range = [0,1]
1931 0BF92 94A  =MEM    ?C=0   S              No arguments?
1932 0BF95 B3           GOYES  MEM10
1933 0BF97 8E00         GOSUBL =PDEV1         Calculate Port#
             00
1934 0BF9D 8F00         GOSBVL =SAVD1         Save D1 in F-R0-1
             000
1935 0BFA4 8E00         GOSUBL =ROMF-         Find Port entry in table
             00
1936 0BFAA 4CD          GOC    MEMeDV
1937 0BFAD 137          CD1EX                 D1 at File chain start
1938 0BFB0 8E00         GOSUBL =EOFLC+        D1 at File chain end
             00
1939 0BFB6 8E00         GOSUBL =LSTADR
             00
1940 0BFBC 171          D1=D1+ 2              Point past zero byte
1941 0BFBF 133          AD1EX
1942 0BFC2 E2           C=C-A  A              #Nibs left
1943 0BFC4 DA           A=C    A
1944 0BFC6 8E00         GOSUBL =rstd1         Restore D1 from F-R0-1
             00
```

```
1945 0BFCC 6A6F          GOTO   LEN10
1946
1947 0BFD0 136    MEM10  CDOEX
1948 0BFD3 D7            D=C    A
1949 0BFD5 1B00          DO=(5) =AVMEMS
           000
1950 0BFDC 146           C=DATO A
1951 0BFDF 164           DO=DO+ 5
1952 0BFE2 AD0           A=0    M
1953 0BFE5 142           A=DATO A
1954 0BFE8 EA            A=A-C  A              COMPUTE FREE BYTES ON STACK
1955 0BFEA D2            C=0    A
1956 0BFEC 314D          LC(2)  =LEEWAY
1957 0BFF0 EA            A=A-C  A              SUBTRACT LEEWAY
1958 0BFF2 540           GONC   MEM20
1959 0BFF5 D0            A=0    A
1960 0BFF7 81C    MEM20  ASRB
1961 0BFFA 6DBE          GOTO   HDFRT2         CONVERT TO DECIMAL, RETURN
1962               ****************************************************************
1963 0BFFE 00            NIBHEX 00             Argument count range = [0,0]
1964 0C000 3F00 =PI      LCHEX  0314159265359000   PI (12 digit form)
           0953
           5629
           5141
           30
1965 0C012 653E          GOTO   fn1              RETURN
1966               ****************************************************************
1967 0C016 8442          NIBHEX 84423          Argument count range = [2,3]
           3
1968 0C01B 80DF =POS     P=C    15             ARGCOUNT->P;
1969 0C01F 136           CDOEX
1970 0C022 10B           R3=C                  PC->R3;
1971 0C025 D1            B=0    A              0->SCANSTART;
1972 0C027 892           ?P=    2              ARGCOUNT=2
1973 0C02A 02            GOYES  POS1           THEN GOTO POS1;
1974 0C02C 7CEC          GOSUB  POP1N
1975 0C030 17F           D1=D1+ 16
1976 0C033 737D          GOSUB  fltdh          SIGN->B;
1977 0C037 D8            B=A    A
1978 0C039 4C0           GOC    POS0           0<=SCANSTART<=FFFFF THEN GO POS0;
1979 0C03C D1            B=0    A              0->SCANSTART;
1980 0C03E 821           XM=0
1981 0C041 94D           ?B#0   S              SIGN#0
1982 0C044 60            GOYES  POS1           THEN GOTO POS1;
1983 0C046 CD     POS0   B=B-1  A              2*(SCANSTART-1)->SCANSTART;
1984 0C048 C5            B=B+B  A
1985 0C04A 7AEC   POS1   GOSUB  POP1S          LEN(SEARCH$)
1986 0C04E D6            C=A    A              ->C;
1987 0C050 133           AD1EX
1988 0C053 CA            A=A+C  A              START(SEARCH$)
1989 0C055 131           D1=A                  ->D1-2
1990 0C058 CC            A=A-1  A
1991 0C05A CC            A=A-1  A
1992 0C05C 101           R1=A                  ->R1;
1993 0C05F 75DC          GOSUB  POP1S          LEN(OBJECT$)->A;
```

```
1994 0C063 137          CD1EX
1995 0C066 C2           C=A+C  A
1996 0C068 10A          R2=C   A              START(OBJECT$)->R2;
1997 0C06B E9           C=C-B  A              START(OBJECT$)-SCANSTART
1998 0C06D 137          CD1EX                 ->SCANPT;
1999 0C070 E0           A=A-B  A              LEN(OBJECT$)-SCANSTART->A<0
2000 0C072 466          GOC    POS5           THEN GOTO POS5;
2001 0C075 EA           A=A-C  A              A-LEN(SEARCH$)->A>0
2002 0C077 416          GOC    POS5           THEN GOTO POS5;
2003 0C07A AD1          B=0    M
2004 0C07D D8           B=A    A
2005 0C07F 81D          BSRB                  A/2->SCANINDEX;
2006 0C082 CE           C=C-1  A              LEN(SEARCH$)-1->C<0
2007 0C084 445          GOC    POS5           THEN GOTO POS5;
2008 0C087 80D0         P=C    0              RMD(LEN(SEARCH$)-1,16)->PARTWORD;
2009 0C08B F6           CSR    A              DIV(LEN(SEARCH$)-1,16)
2010 0C08D D7           D=C    A              ->WORDCOUNT;
2011 0C08F 5D2          GONC   POS3           B.E.T.
2012
2013 0C092 D9    POS2   C=B    A
2014 0C094 06           RSTK=C                SCANINDEX->STACK;
2015 0C096 137          CD1EX
2016 0C099 06           RSTK=C                SCANPT->STACK;
2017 0C09B 137          CD1EX
2018 0C09E 161          D0=D0+ 2
2019 0C0A1 171          D1=D1+ 2
2020 0C0A4 DB           C=D    A
2021 0C0A6 D5           B=C    A              WORDCOUNT->B;
2022 0C0A8 8F00         GOSBVL =STREQL        STRINGMATCH  ------+
           000
2023 0C0AF 07           C=RSTK                                  |
2024 0C0B1 135          D1=C                  SCANPT->D1;       |
2025 0C0B4 07           C=RSTK                                  |
2026 0C0B6 D5           B=C    A              SCANINDEX->B;     |
2027 0C0B8 562          GONC   POS6           THEN GOTO POS6; <--+
2028 0C0BB CD           B=B-1  A              SCANINDEX-1->SCANINDEX<0 --+
2029 0C0BD 119   POS3   C=R1                                            |
2030 0C0C0 134          D0=C                  START(SEARCH$)->D0;       |
2031 0C0C3 451          GOC    POS5           THEN GOTO POS5; <----------+
2032 0C0C6 14A          A=DATO B              FIRSTCHAR(SEARCH$)->A;
2033 0C0C9 1C1   POS4   D1=D1- 2              SCANPT-2->SCANPT;
2034 0C0CC 14F          C=DAT1 B              OBJECT$(SCANPT)->C;
2035 0C0CF 962          ?A=C   B              FIRSTCHR(SEARCH$)=OBJECT$(SCANPT)
2036 0C0D2 0C           GOYES  POS2           THEN REPEAT POS2;
2037 0C0D4 CD           B=B-1  A              SCANINDEX-1->SCANINDEX>=0
2038 0C0D6 52F          GONC   POS4           THEN REPEAT POS4;
2039 0C0D9 11A   POS5   C=R2
2040 0C0DC 135          D1=C                  START(OBJECT$)->SCANPT;
2041 0C0DF 11A   POS6   C=R2
2042 0C0E2 AF0          A=0    W
2043 0C0E5 DA           A=C    A
2044 0C0E7 137          CD1EX                 SCANPT->C;
2045 0C0EA 1CF          D1=D1- 16             MTHSTK-16->D1;
2046 0C0ED EA           A=A-C  A              START(OBJECT$)-SCANPT->A;
2047 0C0EF 81C          ASRB                  A/2->A;
```

```
2048 0C0F2 71DD         GOSUB  hdflt          FLOAT(A)
2049 0C0F6 AF6          C=A    W              ->C;
2050 0C0F9 113          A=R3                  PC->A;
2051 0C0FC 6A3A         GOTO   ARYRTN         GOTO FNRTN3
2052                    ************************************************************
2053 0C100 8822         NIBHEX 8822           Argument count range = [2,2]
2054 0C104 7C4C =RMD    GOSUB  MPOP2N
2055 0C108 403          GOC    CLNK6
2056 0C10B 7000         GOSUB  =RMD12
2057 0C10F 647B         GOTO   OUTRES
2058                    ************************************************************
2059 0C113 8822         NIBHEX 8822           Argument count range = [2,2]
2060 0C117 793C =MOD    GOSUB  MPOP2N
2061 0C11B 4D1          GOC    CLNK6
2062 0C11E 7000         GOSUB  =MOD12
2063 0C122 616B         GOTO   OUTRES
2064                    ************************************************************
2065 0C126 8822         NIBHEX 8822           Argument count range = [2,2]
2066 0C12A 762C =REM    GOSUB  MPOP2N
2067 0C12E 4A0          GOC    CLNK6
2068 0C131 7000         GOSUB  =REM12         IEEE xREMy fcn.
2069 0C135 6E4B         GOTO   OUTRES
2070                    ************************************************************
2071 0C139 8C5E  CLNK6  GOLONG CPOLLF
           AF
2072
2073 0C13F 8822         NIBHEX 8822           Argument count range = [2,2]
2074 0C143 7D0C =DIV    GOSUB  MPOP2N
2075 0C147 41F          GOC    CLNK6
2076 0C14A 7000         GOSUB  =IDIV12        integer divide (xDIVy)
2077 0C14E 653B         GOTO   OUTRES
2078                    ************************************************************
2079 0C152 811          NIBHEX 811            Argument count range = [1,1]
2080 0C155      =DEG
2081 0C155 743C =RTD    GOSUB  MPOP1N
2082 0C159 4FD          GOC    CLNK6          Error if complex
2083 0C15C 7000         GOSUB  =SPLITA
2084 0C160 7230         GOSUB  PI/180
2085 0C164 7000         GOSUB  =DV15M
2086 0C168 6B1B         GOTO   OUTRES
2087                    ************************************************************
2088 0C16C 811          NIBHEX 811            Argument count range = [1,1]
2089 0C16F 7A1C =SQR    GOSUB  MPOP1N          POP ARGUMENT
2090 0C173 45C          GOC    CLNK6
2091 0C176 7000         GOSUB  =SQR12
2092 0C17A 690B         GOTO   OUTRES
2093                    ************************************************************
2094 0C17E 811          NIBHEX 811            Argument count range = [1,1]
2095 0C181      =RAD
2096 0C181 780C =DTR    GOSUB  MPOP1N
2097 0C185 43B          GOC    CLNK6          Error if complex
2098 0C188 7000         GOSUB  =SPLITA
2099 0C18C 7600         GOSUB  PI/180
2100 0C190 8CA1         GOLONG uMP15
           5F
```

```
2101
2102 0C196 27   =PI/180 P=     7
2103 0C198 8E00         GOSUBL =GETCON
           00
2104 0C19E AF7          D=C    W
2105 0C1A1 AF2          C=0    W
2106 0C1A4 CE           C=C-1  A
2107 0C1A6 CE           C=C-1  A
2108 0C1A8 01           RTN
2109            ****************************************************************
2110            ****************************************************************
2111            **
2112            ** Name:    STRLIT  -  Process String Literal Expression
2113            **
2114            ** Category:   FNEXEC
2115            **
2116            ** Purpose:
2117            **      Process a string literal in an expression.
2118            **
2119            ** Entry:
2120            **      D0 = PC.
2121            **      D1 = Mathstack pointer.
2122            **      Token in B[B].
2123            **      Jumped on " or ' token.
2124            **      P=0.
2125            **
2126            ** Exit:
2127            **      eMEM if insufficient memory to put string on stack.
2128            **      Through EXPR.
2129            **
2130            ** Calls:      None.
2131            **
2132            ** Detail:
2133            **      B[B] contains " or '.  This routine scans the string,
2134            **      putting each character on the stack as it is
2135            **      encountered until a matching closing quote is found.
2136            **      The code then builds a string header and exits to
2137            **      expression execution controller.
2138            **
2139            ** History:
2140            **
2141            **    Date     Programmer             Modification
2142            **  --------   ----------   ---------------------------------
2143            **             SA           Wrote
2144            **  10/14/83   NM           Attempted to document
2145            **
2146            ****************************************************************
2147            ****************************************************************
2148 0C1AA D3   =STRLIT D=0    A
2149 0C1AC AD0          A=0    M
2150 0C1AF 137          CD1EX                  CALCULATE FREE BYTES ON STACK
2151 0C1B2 1F00         D1=(5) =AVMEMS
           000
2152 0C1B9 143          A=DAT1 A
2153 0C1BC 135          D1=C
```

```
2154 0C1BF CE          C=C-1  A
2155 0C1C1 EA          A=A-C  A              ANY ROOM ON STACK?
2156 0C1C3 81C         ASRB
2157 0C1C6 451         GOC    SLSTRT         IF SO, GOTO SLSTRT
2158 0C1C9 8C00  STKERR GOLONG =MEMERR
          00
2159 0C1CF E4    PSHCHR A=A+1  A              STACK COLLISION?
2160 0C1D1 47F         GOC    STKERR         IF SO, GOTO STKERR
2161 0C1D4 1C1         D1=D1- 2              PUSH CURRENT CHARACTER ON STACK
2162 0C1D7 14D         DAT1=C B
2163 0C1DA E7          D=D+1  A              INCREMENT LENGTH COUNT
2164 0C1DC 14E   SLSTRT C=DAT0 B              READ CHARACTER
2165 0C1DF 161         D0=D0+ 2              POINT TO NEXT CHARACTER
2166 0C1E2 965         ?B#C   B              TERMINATOR?
2167 0C1E5 AE          GOYES  PSHCHR         IF NOT, REPEAT PSHCHR
2168 0C1E7 DB          C=D    A
2169 0C1E9 C6          C=C+C  A              CONVERT BYTES TO NIBBLES
2170 0C1EB BF2         CSL    W              CONSTRUCT STRING HEADER
2171 0C1EE BF2         CSL    W
2172 0C1F1 30F         LCHEX  F
2173 0C1F4 635C        GOTO   fn1            RETURN
2174            ********************************************************************
2175            ********************************************************************
2176            **
2177            ** Name:    SUB$    -  Process Substring
2178            **
2179            ** Category:   FNEXEC
2180            **
2181            ** Purpose:
2182            **      Process substring specification.
2183            **
2184            ** Entry:
2185            **      P=0.
2186            **      D0 = PC.
2187            **      D1 = Mathstack pointer.
2188            **      Jumped on substring token.
2189            **
2190            ** Exit:
2191            **      Through EXPR.
2192            **
2193            ** Calls:      MOVEDD, POP1N.
2194            **
2195            ** Detail:
2196            **      Substrings are tokenized as:
2197            **        <string expr>
2198            **        <expression>
2199            **        (<expression>, if 2nd substring parm)
2200            **        substring token (w/number parms)
2201            **      SUB$ then peels substring parameter(s) off of stack
2202            **      and modifies string (on stack) into substring.
2203            **
2204            ** History:
2205            **
2206            **    Date     Programmer             Modification
2207            **  --------   ----------   ------------------------------------
```

```
2208            **              SA          Wrote
2209            **  10/14/83    NM          Attempted to document
2210            **
2211            ********************************************************************
2212            ********************************************************************
2213 0C1F8 8812          NIBHEX 8812          Argument count range = [1,2]
2214 0C1FC D3   =SUB$    D=0    A             CREATE DEFAULT SECOND ARGUMENT
2215 0C1FE CF            D=D-1  A
2216 0C200 1564          C=DAT0 S             READ ARGUMENT COUNT
2217 0C204 160           D0=D0+ 1
2218 0C207 A4E           C=C-1  S
2219 0C20A 7E0B  GETARG  GOSUB  POP1N         READ ARGUMENT
2220 0C20E 789B          GOSUB  fltdh         CONVERT TO HEX INTEGER
2221 0C212 4B0           GOC    POPARG        IF POSITIVE, GOTO POPARG
2222 0C215 D0            A=0    A             NEGATIVE ARGUMENT BECOMES ZERO
2223 0C217 94D           ?B#0   S             NEGATIVE ARGUMENT?
2224 0C21A 40            GOYES  POPARG        IF SO, GOTO POPARG
2225 0C21C CC            A=A-1  A             LARGE ARG OR IEEE BECOMES FFFFF
2226 0C21E 17F   POPARG  D1=D1+ 16            POP STACK
2227 0C221 D6            C=A    A             SWAP ARGUMENTS
2228 0C223 DF            CDEX   A
2229 0C225 DA            A=C    A
2230 0C227 A4E           C=C-1  S             ANOTHER ARGUMENT ON STACK?
2231 0C22A 5FD           GONC   GETARG        IF SO, REPEAT GETARG
2232 0C22D 8AB           ?D=0   A             FIRST ARGUMENT ZERO?
2233 0C230 40            GOYES  SUB$10        IF SO, GOTO SUB$10
2234 0C232 CF            D=D-1  A             DECREMENT ARGUMENT
2235 0C234 C4    SUB$10  A=A+A  A
2236 0C236 C7            D=D+D  A
2237 0C238 136           CD0EX                SAVE PC IN D(A)
2238 0C23B DF            CDEX   A
2239 0C23D D5            B=C    A
2240 0C23F 8B0           ?A>B   A             SECOND ARG GREATER THAN FIRST?
2241 0C242 40            GOYES  SUB$20        IF SO, GOTO SUB$20
2242 0C244 D4            A=B    A             SET SECOND ARG EQUAL TO FIRST
2243 0C246 1B00  SUB$20  D0=(5) =F-R0-2
           000
2244 0C24D 140           DAT0=A A             RECORD SECOND ARGUMENT
2245 0C250 184           D0=D0- 5
2246 0C253 144           DAT0=C A             RECORD FIRST ARGUMENT
2247 0C256 184           D0=D0- 5
2248 0C259 176           D1=D1+ 7
2249 0C25C 147           C=DAT1 A
2250 0C25F 144           DAT0=C A             RECORD SOURCE ADDRESS
2251 0C262 1C6           D1=D1- 7
2252 0C265 1577          C=DAT1 W             READ STRING HEADER
2253 0C269 B06           C=C+1  P             STRING?
2254 0C26C 527           GONC   SUB$60        IF NOT, GOTO SUB$60
2255 0C26F 30C           LCHEX  C             ASSUME NONEXISTENT STRING ARRAY
2256 0C272 816           CSRC
2257 0C275 A06           C=C+C  P             ASSUMPTION PROBABLY CORRECT?
2258 0C278 450           GOC    SUB$30        IF SO, GOTO SUB$30
2259 0C27B B46           C=C+1  S             ASSUME SIMPLE STRING VARIABLE
2260 0C27E 90E   SUB$30  ?C#0   P             ASSUMPTION CORRECT?
2261 0C281 E5            GOYES  SUB$60        IF NOT, GOTO SUB$60
```

```
2262 0C283 AC5          B=C    S
2263 0C286 BF6          CSR    W              ACTUAL LENGTH IN C(A)
2264 0C289 8B2          ?A<C   A              SECOND ARG BEYOND ACTUAL STRING?
2265 0C28C 40           GOYES  SUB$40          IF NOT, GOTO SUB$40
2266 0C28E DA           A=C    A              SET TO ACTUAL STRING END
2267 0C290 8B5   SUB$40 ?B<C   A              FIRST ARG BEYOND ACTUAL STRING?
2268 0C293 40           GOYES  SUB$50          IF NOT, GOTO SUB$50
2269 0C295 D5           B=C    A              SET TO ACTUAL STRING END
2270 0C297 E0    SUB$50 A=A-B  A              COMPUTE LENGTH OF SUBSTRING
2271 0C299 17F          D1=D1+ 16
2272 0C29C 133          AD1EX
2273 0C29F ED           B=C-B  A              COMPUTE OFFSET TO STRT OF SUBSTR
2274 0C2A1 C2           C=A+C  A              COMPUTE ADDR OF START OF STRING
2275 0C2A3 C0           A=A+B  A              COMPUTE ADDR OF START OF SUBSTR
2276 0C2A5 137          CD1EX
2277 0C2A8 D5           B=C    A              LEAVE SUBSTRING LENGTH IN B(A)
2278 0C2AA 8F00         GOSBVL =MOVEDD        MOVE SUBSTR TO NEW TOP OF STACK
          000
2279 0C2B1 D9           C=B    A              CONSTRUCT STRING HEADER
2280 0C2B3 BF2          CSL    W
2281 0C2B6 BF2          CSL    W
2282 0C2B9 30F          LCHEX  F
2283 0C2BC 1CF          D1=D1- 16
2284 0C2BF 1557         DAT1=C W              WRITE STRING HEADER TO STACK
2285 0C2C3 1B00         D0=(5) =F-R0-0
          000
2286 0C2CA 15EE         C=DAT0 15             STORE ADDR INFO IN B-REGISTER
2287 0C2CE AB5          B=C    X
2288 0C2D1 AD5          B=C    M
2289 0C2D4 DB    DEXPR  C=D    A
2290 0C2D6 134   CEXPR  D0=C
2291 0C2D9 8CC3  expr1  GOLONG  expr          RETURN
          6F
2292 0C2DF 653A  SUB$60 GOTO   CNFLCT
2293             ********************************************************************
2294             ********************************************************************
2295             **
2296             ** Name:    XFN     -  Process External Functions
2297             **
2298             ** Category:   FNEXEC
2299             **
2300             ** Purpose:
2301             **      Process XFNs in expression.
2302             **
2303             ** Entry:
2304             **      P=0.
2305             **      D0 = PC.
2306             **      D1 = Mathstack pointer.
2307             **      Jumped on XFN token.
2308             **
2309             ** Exit:
2310             **      Exits to function requested.
2311             **
2312             ** Calls:      XMTADR.
2313             **
```

```
2314                ** Detail:
2315                **      Looks for Xfunction table and then looks for requested
2316                **      entry in table.  If not found then errors with eXFNNF.
2317                **      Otherwise loads address of function execute and jumps
2318                **      to it.
2319                **
2320                ** History:
2321                **
2322                **    Date      Programmer              Modification
2323                **  --------    ----------    ----------------------------------
2324                **              SA            Wrote
2325                **  10/14/83    NM            Attempted to document
2326                **
2327                ****************************************************************
2328                ****************************************************************
2329 0C2E3 0F             NIBHEX 0F              Argument count range = [0,15]
2330 0C2E5 142  =XFN      A=DAT0 A
2331 0C2E8 8E00           GOSUBL =XMTADR
             00
2332 0C2EE 4E2           GOC    NOXFN
2333 0C2F1 C9            C=C+B  A
2334 0C2F3 A35           B=B+B  X
2335 0C2F6 A35           B=B+B  X
2336 0C2F9 A35           B=B+B  X
2337 0C2FC C9            C=C+B  A
2338 0C2FE AB8           B=A    X               copy token into B(X)
2339 0C301 136           CDOEX
2340 0C304 162           D0=D0+ 3
2341 0C307 142           A=DAT0 A
2342 0C30A 136           CDOEX
2343 0C30D C2            C=C+A  A
2344 0C30F 06            RSTK=C
2345 0C311 163           D0=D0+ 4
2346 0C314 1564          C=DAT0 S
2347 0C318 160           D0=D0+ 1
2348 0C31B 03            RTNCC
2349 0C31D 20   =NOXFN   P=     0
2350 0C31F 3100          LC(2)  =eXFNNF
2351 0C323 6BFB          GOTO   mferr
2352 0C327               END
```

```
 2xRGNM  Abs   47534 #0B9AE -   603   826   855
=ABS     Abs   47369 #0B909 -   514
=ACOS    Abs   48617 #0BDE9 -  1642
 ACOS12  Ext               -  1644
 ACfrt2  Abs   48830 #0BEBE -  1752  1762
 ADDRSS  Ext               -   790   894
=AND     Abs   47223 #0B877 -   428
=ANGLE   Abs   48658 #0BE12 -  1654
 ARG12   Ext               -  1656
=ARGERR  Abs   48921 #0BF19 -  1891
=ARRAY   Abs   47837 #0BADD -   790
 ARYADR  Abs   47536 #0B9B0 -   604   815   845
 ARYRTN  Abs   47927 #0BB37 -   817   702   729   733  2051
=ASIN    Abs   48591 #0BDCF -  1634
 ASIN12  Ext               -  1636
=ATAN    Abs   48637 #0BDFD -  1648
 ATAN12  Ext               -  1650
 AVMEME  Ext               -  1002
 AVMEMS  Ext               -  1949  2151
 B9605B  Ext               -   692
=BABBAJ  Abs   47206 #0B866 -   389    81   350   433   435
 BASE    Ext               -   649
=BIG     Abs   46919 #0B747 -   221
=BIG+    Abs   46916 #0B744 -   220   270
=BOOLE   Abs   47203 #0B863 -   388
 CEXPR   Abs   49878 #0C2D6 -  2290
=CHR$    Abs   48678 #0BE26 -  1660
 CHR$1   Abs   48701 #0BE3D -  1667  1664
=CLASS   Abs   46881 #0B721 -   168
 CLASSA  Ext               -   170
 CLNK1   Abs   46774 #0B6B6 -   103    66    72    78    85    91    96   107
                           -   134   169
 CLNK2   Abs   47039 #0B7BF -   299   348
 CLNK3   Abs   47362 #0B902 -   511   440   448   494   515   556
 CLNK4   Abs   48716 #0BE4C -  1672  1635  1643  1649  1655  1676  1682  1688
 CLNK5   Abs   48987 #0BF5B -  1913  1834  1840  1909  1917  1923
 CLNK6   Abs   49465 #0C139 -  2071  2055  2061  2067  2075  2082  2090  2097
=CMPLX   Abs   48156 #0BC1C -   986
 CMPLX1  Abs   48165 #0BC25 -   990   986
=CNFLCT  Abs   48405 #0BD15 -  1263   173   831  1212  1236  1393  2292
=COMPAR  Abs   47045 #0B7C5 -   333
=CONCAT  Abs   46799 #0B6CF -   112
=COS     Abs   48743 #0BE67 -  1681
 COS12   Ext               -  1683
 CPLX    Abs   48009 #0BB89 -   849   811
=CPOLL   Abs   48182 #0BC36 -   996   991
=CPOLLF  Abs   48160 #0BC20 -   988   103   299   511  1672  1913  2071
=CPOLLR  Abs   48167 #0BC27 -   991   861
=CSETUP  Abs   48035 #0BBA3 -   859
 CSHORT  Abs   48023 #0BB97 -   855   853
 D1AEXP  Abs   47379 #0B913 -   517    74  1905
 DATERR  Abs   46898 #0B732 -   173
=DEG     Abs   49493 #0C155 -  2080
 DEXPR   Abs   49876 #0C2D4 -  2289   130
 DIM10   Ext               -   696
```

```
=DIV     Abs   49475 #0C143 -  2074
=DIVIDE  Abs   46725 #0B685 -    90
 DMA10   Abs   48109 #0BBED -   913   911
 DMA100  Abs   48128 #0BC00 -   920   896
 DMA20   Abs   48119 #0BBF7 -   916   927   933
=DMARRY  Abs   48050 #0BBB2 -   894
 DMPRES  Abs   47217 #0B871 -   393   390   509
 DOOFUS  Abs   47965 #0BB5D -   831   796   900
=DTR     Abs   49537 #0C181 -  2096
 DV15M   Ext                -  2085
 DV2-12  Ext                -    92
 EOFLC+  Ext                -  1938
=EPS     Abs   47009 #0B7A1 -   288
 ERR#    Ext                -  1738
=ERRL    Abs   48848 #0BED0 -  1757
 ERRL#   Ext                -  1757
=ERRN    Abs   48782 #0BE8E -  1738
 EX-112  Ext                -  1841
 EX12    Ext                -   135
 EXAB1   Ext                -  1842
=EXOR    Abs   47269 #0B8A5 -   446
 EXOR10  Abs   47292 #0B8BC -   454   452
 EXOR20  Abs   47305 #0B8C9 -   458   456
=EXP     Abs   48878 #0BEEE -  1833
 EXP12   Ext                -  1835
=EXPM1   Abs   48898 #0BF02 -  1839
=EXPON   Abs   46861 #0B70D -   133
 EXPR    Ext                -   518
 F-R0-0  Ext                -  2285
 F-R0-2  Ext                -  2243
 F-R1-0  Ext                -   659   802
 F-R1-3  Ext                -   654
 FAKE    Ext                -   739
 FLGREG  Ext                -  1583
 FLOATA  Ext                -  1761
 FLTDH   Ext                -  1538
 FN1     Abs   47033 #0B7B9 -   297   271   285   289   293  1670
 FN4     Abs   48608 #0BDE0 -  1639   171   393  1051  1666
 FNRTN1  Ext                -   297
 FNRTN2  Ext                -  1753
 FNRTN3  Ext                -   817
 FNRTN4  Ext                -  1639
 FPOLL   Ext                -  1006
 FUNCDO  Ext                -   998  1009
 GETARG  Abs   49674 #0C20A -  2219  2231
 GETCON  Ext                -  2103
 HDFLT   Ext                -  1754
 HDFRT2  Abs   48824 #0BEB8 -  1750  1961
=HUGE    Abs   46941 #0B75D -   228   292
=HUGE20  Abs   46944 #0B760 -   229
 IDIV12  Ext                -  2076
=INF     Abs   47018 #0B7AA -   292
 INT     Abs   47945 #0BB49 -   824   822
 INTSHT  Ext                -   829
=INVLUT  Abs   46651 #0B63B -    65
```

```
 INVNaN  Ext                    -  1018
=INVRES  Abs   48248 #0BC78    -  1018   715
 LEEWAY  Abs     212 #000D4    -    12  1956
=LEN     Abs   48936 #0BF28    -  1896
 LEN10   Abs   48951 #0BF37    -  1901  1945
=LEN20   Abs   48959 #0BF3F    -  1904
 LGCTST  Abs   47240 #0B888    -   434   443   460
 LGT12   Ext                    -  1918
 LN1+12  Ext                    -  1924
 LN12    Ext                    -  1910
=LOG     Abs   48970 #0BF4A    -  1908
=LOG10   Abs   48994 #0BF62    -  1916
=LOGIC   Abs   47209 #0B869    -   390
=LOGP1   Abs   49014 #0BF76    -  1922
 LSTADR  Ext                    -  1939
=MAXRL   Abs   46958 #0B76E    -   270
=MEM     Abs   49042 #0BF92    -  1931
 MEM10   Abs   49104 #0BFD0    -  1947  1932
 MEM20   Abs   49143 #0BFF7    -  1960  1958
 MEMERR  Ext                    -  2158
 MEMeDV  Abs   49031 #0BF87    -  1927  1936
 MFERR   Ext                    -  1893
=MINRL   Abs   46967 #0B777    -   274
=MINUS   Abs   46672 #0B650    -    71
=MOD     Abs   49431 #0C117    -  2060
 MOD12   Ext                    -  2062
 MOVEDD  Ext                    -  2278
 MOved3  Ext                    -   123
 MP2-12  Ext                    -    86
 MP2-15  Ext                    -   101
=MPOP1N  Abs   48525 #0BD8D    -  1529   133  1634  1642  1648  1660  1675  1681
                                  1687  1833  1839  1908  1916  1922  2081  2089
                                  2096
=MPOP2N  Abs   48468 #0BD54    -  1439    65    84    90    95   106   343  1654
                                  2054  2060  2066  2074
 MSN15   Ext                    -   497
=MULTPY  Abs   46706 #0B672    -    84
=NAN     Abs   47027 #0B7B3    -   296
 NEWVEC  Abs   47609 #0B9F9    -   635   695   920
=NOT     Abs   46688 #0B660    -    77
=NOXFN   Abs   49949 #0C31D    -  2349   994  2332
 NOXLNK  Abs   48178 #0BC32    -   994   992
 NWVC1   Abs   47633 #0BA11    -   648   646
=OR      Abs   47249 #0B891    -   438
=OUTRES  Abs   48260 #0BC84    -  1050    68    87    93   102   136  1637  1645
                                  1651  1657  1678  1684  1690  1836  1843  1911
                                  1919  1925  2057  2063  2069  2077  2086  2092
 PDEV1   Ext                    -  1933
=PERCNT  Abs   46744 #0B698    -    95
=PI      Abs   49152 #0C000    -  1964
=PI/180  Abs   49558 #0C196    -  2102  2084  2099
=PLUS    Abs   46780 #0B6BC    -   106
=POP1N   Abs   48412 #0BD1C    -  1303    71   168   493   514   553   565   597
                                  1530  1974  2219
=POP1N+  Abs   48529 #0BD91    -  1530    77
```

```
=POP1S   Abs   48440 #OBD38 -  1388   112   357  1896  1985  1993
 POP1S1  Abs   48444 #OBD3C -  1389   120
 POP1S2  Abs   48457 #OBD49 -  1394   351
=POP2N   Abs   48268 #OBC8C -  1195  1440
=POP2N+  Abs   48472 #OBD58 -  1440   428   438   446
 POP2NA  Abs   48498 #OBD72 -  1447  1444
 POP2NB  Abs   48517 #OBD85 -  1452  1449
 POPARG  Abs   49694 #OC21E -  2226  2221  2224
=POS     Abs   49179 #OC01B -  1968
 POS0    Abs   49222 #OC046 -  1983  1978
 POS1    Abs   49226 #OC04A -  1985  1973  1982
 POS2    Abs   49298 #OC092 -  2013  2036
 POS3    Abs   49341 #OC0BD -  2029  2011
 POS4    Abs   49353 #OC0C9 -  2033  2038
 POS5    Abs   49369 #OC0D9 -  2039  2000  2002  2007  2031
 POS6    Abs   49375 #OC0DF -  2041  2027
 PSHCHR  Abs   49615 #OC1CF -  2159  2167
=RAD     Abs   49537 #OC181 -  2095
 RDATTY  Ext               -  1263
 RECADR  Ext               -   914
=RED     Abs   49450 #OC12A -  2066
 REGNUM  Abs   47418 #OB93A -   559   697   800
 REM12   Ext               -  2068
=RES     Abs   47391 #OB91F -   551
 RESREG  Ext               -   552
 REV$    Ext               -  1346
=REVPOP  Abs   48433 #OBD31 -  1346
 RGNM1   Abs   47434 #OB94A -   565   583   588
 RGNM2   Abs   47453 #OB95D -   571   568
 RGNM3   Abs   47476 #OB974 -   580   577
 RGNM4   Abs   47480 #OB978 -   583   563
 RGNM5   Abs   47519 #OB99F -   596   584
=RMD     Abs   49412 #OC104 -  2054
 RMD12   Ext               -  2056
 ROMF-   Ext               -  1935
 RSLV10  Abs   48301 #OBCAD -  1208  1201
 RSLV20  Abs   48359 #OBCE7 -  1228  1205
 RSLV30  Abs   48371 #OBCF3 -  1232  1221  1309
=RTD     Abs   49493 #OC155 -  2081
 RTNNAN  Abs   47772 #OBA9C -   710   698   805
 SAVD1   Ext               -  1934
 SAVSUB  Abs   47667 #OBA33 -   659   699   921
 SC10    Abs   47153 #OB831 -   372   366   369
 SC20    Abs   47182 #OB84E -   381   379
 SC30    Abs   47186 #OB852 -   382   376
=SGN     Abs   47320 #OB8D8 -   493
 SGN20   Abs   47345 #OB8F1 -   503   499
 SGN30   Abs   47358 #OB8FE -   509   501   506
 SHORT   Abs   47933 #OBB3D -   819   813
=SIGCHK  Abs   48536 #OBD98 -  1532
 SIGNAN  Ext               -   296
 SIGTST  Ext               -  1443  1448  1532
=SIN     Abs   48723 #OBE53 -  1675
 SIN12   Ext               -  1677
 SLARF   Abs   47255 #OB897 -   440   430
```

```
 SLSTRT  Abs   49628 #0C1DC -  2164  2157
=SMALL   Abs   46902 #0B736 -   216   288
 SPLITA  Ext                -  2083  2098
 SPLTAC  Ext                -    97
=SQR     Abs   49519 #0C16F -  2089
 SQR12   Ext                -  2091
 STKERR  Abs   49609 #0C1C9 -  2158  2160
 STRCMP  Abs   47100 #0B7FC -   351   340
 STREQL  Ext                -  2022
 STRING  Abs   47969 #0BB61 -   836   808
=STRLIT  Abs   49578 #0C1AA -  2148
 STRRCL  Ext                -   847
 STRTST  Ext                -   374
=SUB$    Abs   49660 #0C1FC -  2214
 SUB$10  Abs   49716 #0C234 -  2235  2233
 SUB$20  Abs   49734 #0C246 -  2243  2241
 SUB$30  Abs   49790 #0C27E -  2260  2258
 SUB$40  Abs   49808 #0C290 -  2267  2265
 SUB$50  Abs   49815 #0C297 -  2270  2268
 SUB$60  Abs   49887 #0C2DF -  2292  2254  2261
=TAN     Abs   48763 #0BE7B -  1687
 TAN12   Ext                -  1689
 TOFN4   Abs   46894 #0B72E -   171   109
 UNDECL  Abs   47741 #0BA7D -   695   792
 VCCK1   Abs   47558 #0B9C6 -   616   627
 VECCHK  Abs   47569 #0B9D1 -   620   795   899
=XFN     Abs   49893 #0C2E5 -  2330
 XMTADR  Ext                -  2331
 YX2-12  Ext                -    67
 a-mult  Ext                -   591   844  1745
 cRCL    Abs     103 #00067 -    12   860
 csrc5   Ext                -   684
 eDVCNF  Ext                -  1927
 eIVARG  Ext                -  1892
 eSUBSC  Ext                -   714
 eXFNNF  Ext                -  2350
 expr    Abs   47383 #0B917 -   518   993  2291
 expr1   Abs   49881 #0C2D9 -  2291
=fltdh   Abs   48554 #0BDAA -  1538   567  1661  1976  2220
 fn1     Abs   48712 #0BE48 -  1670   557  1965  2173
 fnrtn2  Abs   48833 #0BEC1 -  1753   918
=hdflt   Abs   48839 #0BEC7 -  1754  1750  1904  2048
 mferr   Abs   48927 #0BF1F -  1893  1928  2351
 pCMPLX  Abs      56 #00038 -    12  1007
 rstd1   Ext                -  1944
 uAD12   Ext                -   108
=uMODES  Abs   48561 #0BDB1 -  1581  1439  1529
 uMP15   Abs   46764 #0B6AC -   101    98  2100
 uRES12  Ext                -  1019
 uTEST   Ext                -   349
 ures12  Abs   48254 #0BC7E -  1019   284  1050  1445  1450  1534
```

Input Parameters

  Source file name is AB&FCN::MS

  Listing file name is AB/FCN:TI:ML::-1

  Object file name is AB%FCN:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
1            *        J  TTTTT   &     M   M  TTTTT  H   H
2            *        J    T    & &    MM MM    T    H   H
3            *        J    T    & &    M M M    T    H   H
4            *        J    T     &     M M M    T    H H H
5            *        J    T    & & &  M   M    T    H   H
6            *    J   J    T    &  &   M   M    T    H   H
7            *     JJJ     T     && &  M   M    T    H   H
8
9                      TITLE  Math Routines - Part 1 <831213.1007>
10 0C327               ABS    #0C327
11           ************************************************
12           ***   MATH ROUTINES                          ***
13           ***                                          ***
14           ***  INPUT:                                  ***
15           ***                                          ***
16           ***     FOR BINARY FUNCTIONS-                ***
17           ***                                          ***
18           ***       IF NUMBER IS 2-12,                 ***
19           ***          THEN FORM IS:                   ***
20           ***             A HAS 12 DIGIT FORM          ***
21           ***             C HAS 12 DIGIT FORM          ***
22           ***       IF NUMBER IS 1-12,                 ***
23           ***          THEN FORM IS:                   ***
24           ***             A HAS SIGN AND EXP           ***
25           ***             B HAS 15 DIGIT MANTISSA      ***
26           ***             C HAS 12 DIGIT FORM          ***
27           ***       IF NUMBER IS 2-15,                 ***
28           ***          THEN FORM IS:                   ***
29           ***             A HAS SIGN AND EXP           ***
30           ***             B HAS 15 DIGIT MANTISSA      ***
31           ***             C HAS SIGN AND EXP           ***
32           ***             D HAS 15 DIGIT MANTISSA      ***
33           ***                                          ***
34           ***    FOR UNARY FUNCTIONS-                  ***
35           ***                                          ***
36           ***      IF NUMBER IS 12,                    ***
37           ***         THEN FORM IS:                    ***
38           ***            C HAS 12 DIGIT FORM           ***
39           ***      IF NUMBER IS 15,                    ***
40           ***         THEN FORM IS:                    ***
41           ***            A HAS SIGN AND EXP            ***
42           ***            B HAS 15 DIGIT MANTISSA       ***
43           ***                                          ***
44           ***                                          ***
45           ***  OUTPUT:                                 ***
46           ***            A HAS SIGN AND EXP            ***
47           ***            B HAS 15 DIGIT MANTISSA       ***
48           ***                                          ***
49           ***            SB=0: MANTISSA IN B EXACT     ***
50           ***            SB=1: MANTISSA IN B INEXACT   ***
51           ***                                          ***
52           ***                                          ***
53           ************************************************
54
55
```

```
56
57
58
59
60          ****************************************************************
61          ***                                                           *
62          *      STATUS BITS                                          *
63          *                                                           *
64          =sINFRD EQU    10           RND TO +/- INF                  *
65          =sNEGRD EQU    11           RND TO -INF OR TO 0             *
66          =sXCPT  EQU    4            GLOBAL XCPT FLAG                *
67          =sINX   EQU    5            GLOBAL INEXACT FLAG             *
68          =sIX    EQU    7               .                            *
69           sUN    EQU    8               . local flags in             *
70           sOV    EQU    9               .  uRES12                    *
71           sDZ    EQU    10              .                            *
72           sIV    EQU    11              .                            *
73           sY=INF EQU    8            (y=inf flag used in YTOX)       *
74           s=SGNS EQU    9            used in remainder fcns.
75           NOTMOD EQU    8            USED IN REMPWD ONLY (for xMODinf)
76                                                                      *
77          *    XCPTIONS FOR P                                        *
78          *                                                          *
79          =OKP    EQU    0           INEXACT (needs to stay 0 for* HTRAP
80          =UNP    EQU    1           UNDERFLOW                      *
81          =OVP    EQU    2           OVERFLOW                       *
82          =DZP    EQU    3           ZERO DIV.                      *
83          =IVP    EQU    4           INVALID                        *
84                                                                    *
85                                                                    *
86          =EFIELD EQU    0           PTR POSITION FOR BYTE ERROR    *
87                                        FIELD IN NaN .              *
88                                                                    *
89                                                                    *
90          *    OPERATIONS FOR P                                     *
91          *                                                         *
92          +       EQU    0           P=0 => P='+'                   *
93          x       EQU    1                                          *
94          /       EQU    2                                          *
95                                                                    *
96                                                                    *
97          ****************************************************************
98
99
100
101
102         ********************************************************************
103         ********************************************************************
104         **
105         ** Name:(S) ADDONE  -  Add One
106         ** Name:(S) SUBONE  -  Subtract One
107         **
108         ** Category:   MATH
109         **
110         ** Purpose:
```

```
                **      To compute  X+1 & X-1 for X an internal number.
                **
                ** Entry:
                **      Standard floating point math input with (A,B)=X.
                **      sINFRD(s10)&sNEGRD(s11),rounding modes, are consulted
                **        only if X+1=0 (or X-1=0) in which case the result may
                **        be +0 or -0 depending on the mode.(see AD15s)
                **
                ** Exit:
                **      Standard floating point math output.
                **
                ** Calls:      Goes to AD15s .
                **
                ** Uses.......
                **  Inclusive: P; A,B,C,D;
                **                HD.ST.[SB];
                **
                ** Stk lvls:   0
                **
                ** NOTE:
                **      Can raise no XM=1 xcption . (clrs SB but not XM)
                **
                ***************************************************************
                ***************************************************************
0C327 AF2       =SUBONE C=0    W              doesn't init SB & XM  (sticky only)
0C32A BCE               C=-C-1 S
0C32D 550               GONC   SUBON1         "goto"

0C330 AF2       =ADDONE C=0    W              DOESN'T INIT SB & XM (sticky only)
0C333 2E         SUBON1 P=     14
0C335 AF3               D=0    W
0C338 B07               D=D+1  P
0C33B 5D2               GONC   AD15s          "GOTO"
                ***************************************************************
                ***************************************************************
                **
                ** Name:(S) 1/X15   -  1/X
                **
                ** Category:   MATH
                **
                ** Purpose:
                **      To compute  1/x
                **
                ** Entry:
                **      Standard floating point math input.
                **
                ** Exit:
                **      Standard floating point math output.
                **
                ** Uses.......
                **  Inclusive: P; A,B,C,D;
                **             HD.ST.[SB,XM];
                **
                ** Stk lvls:   0
                **
```

```
166             ** NOTE:
167             **      Goes to DV15S  (divides 1 by x )
168             **
169             ********************************************************************
170             ********************************************************************
171 0C33E 821   =1/X15  XM=0
172 0C341 822          SB=0
173
174 0C344 AF2   =1/X15S C=0    W
175 0C347 AF3          D=0    W
176 0C34A 2E           P=     14
177 0C34C B07          D=D+1  P
178 0C34F AFD   =X/Y15  BCEX   W
179 0C352 AFF          CDEX   W
180 0C355 AFD          BCEX   W
181 0C358 AFE          ACEX   W
182 0C35B 6651         GOTO   DV15S
```

```
183                     EJECT
184                 ****************************************************************
185                 ****************************************************************
186                 **
187                 ** Name:(S) AD2-15  -  Add two 15 digit forms
188                 ** Name:(S) AD2-12  -  Add two 12 digit forms
189                 ** Name:(S) ADDF    -  Add for finite args only
190                 ** Name:(S) AD15M   -  Add according to modes
191                 ** Name:(S) AD15s   -  Add with XM sticky
192                 **
193                 ** Category:   MATH
194                 **
195                 ** Purpose:
196                 **      To compute  the sum x+y .
197                 **
198                 ** Entry:
199                 **      Standard floating point math input.
200                 **      AD2-15 assumes NOT(round to -Inf)  (i.e. x+(-x)=+0)
201                 **      AD15s has rounding mode inputs (SB cleared inside)
202                 **      s10 & s11 set ==> rnd to -inf (i.e. x+(-x) = -0 )
203                 **       else x+(-x) = +0.
204                 **
205                 **      CODE:  =AD2-12  GOSUB  SPLTAC
206                 **      ----   =AD2-15  ST=0   sNEGRD   (s11) NO round to NEG.
207                 **             =AD15M   XM=0
208                 **             =AD15s   SB=0            (add uses SB for result!)
209                 **
210                 **
211                 ** Exit:
212                 **      Standard floating point math output.
213                 **      XM=1 implies Inf+(-Inf)
214                 **
215                 ** Calls:      (none)
216                 **
217                 ** Uses.......
218                 **  Inclusive: P; A,B,C,D;  ST.[s11 for AD2-15 only];
219                 **             HD.ST.[SB,XM];
220                 **
221                 ** Stk lvls:   0
222                 **
223                 ** NOTE:
224                 **      The main entry AD2-15 forces rnd to nearest
225                 **       (same result except for rnd to -inf).
226                 **      Results are truncated. (e.g. 1 - 1E-100 -->
227                 **       .999999999999999 with SB=1 ! )
228                 **
229                 ****************************************************************
230                 ****************************************************************
231 0C35F          =AD2-12
232 0C35F 71D5             GOSUB  SPLTAC
233 0C363 84B   =AD2-15 ST=0   sNEGRD         indicates DON'T RND TO -INF
234
235 0C366          =AD15M
236 0C366 821              XM=0
237 0C369 822   =AD15s  SB=0                  RESET STICKY BIT
```

```
238 0C36C 20          P=     +              P='+'
239 0C36E 6462        GOTO   PWEEDS         "GOSUB" (RTN TO ADDF)
240           *       .                      .
241           *       .                      .
242           *       .                      .
243 0C372     =ADDF
244 0C372 822         SB=0                  (clear again for ARG (trig))
245 0C375 AFD         BCEX   W              (MANTS IN C&D)
246 0C378 AC2         C=0    S
247 0C37B AC3         D=0    S
248 0C37E 24          P=     4
249 0C380 B04         A=A+1  P
250 0C383 B04         A=A+1  P
251 0C386 B04         A=A+1  P              ADD 50000 TO EACH EXP
252 0C389 B04         A=A+1  P              TO MAKE BOTH POSITIVE
253 0C38C B04         A=A+1  P
254 0C38F B05         B=B+1  P
255 0C392 B05         B=B+1  P
256 0C395 B05         B=B+1  P
257 0C398 B05         B=B+1  P
258 0C39B B05         B=B+1  P
259 0C39E 97B         ?D=0   W              ZERO ARGUMENT?
260 0C3A1 E4          GOYES  ADD60          YES
261           *  PUT SMALLER OF ARGUMENTS IN B & D
262 0C3A3 AFC  ADD20  ABEX   W              EXCHANGE MANTISSAS
263 0C3A6 AFF         CDEX   W              EXCHANGE SIGNS,EXPS
264 0C3A9 97B         ?D=0   W              OTHER ARG ZERO?
265 0C3AC 34          GOYES  ADD60          YES
266 0C3AE 8B0         ?A>B   A              A HAS STRICTLY LARGER EXP?
267 0C3B1 32          GOYES  ADD30          YES. LARGER ARG IS IN A & C
268 0C3B3 8B4         ?A<B   A              A HAS STRICTLY SMALLER EXP?
269 0C3B6 DE          GOYES  ADD20          YES. EXCHANGE
270 0C3B8 9FF         ?C>=D  W              EXPS EQUAL. C>=D (MAN) ?
271 0C3BB 32          GOYES  ADD40          YES
272 0C3BD AFC         ABEX   W              NO.EXCHANGE MANTISSAS
273 0C3C0 AFF         CDEX   W              EXCHANGE SIGNS,EXPS
274 0C3C3 5A1         GONC   ADD40          BET
275
276
277           **********************************************
278
279           *  X&Y ARE FINITE
280 0C3C6     X&Y#S
281
282 0C3C6     MATHR                        MATH RETURN for PWEEDS (TO AVOID
283 0C3C6 890         ?P=    +                 *  SUBR LEVELS)
284 0C3C9 9A          GOYES  ADDF              *
285                                            *
286 0C3CB 891         ?P=    x                 *
287 0C3CE 87          GOYES  MULTF             *
288                                            *
289 0C3D0 67E0        GOTO   DIVF              *
290           ************************************************************
291
292
```

```
 293 0C3D4 940   ADD30  ?A=B   S             LIKE SIGNS?
 294 0C3D7 70           GOYES  ADD40         YES
 295 0C3D9 BF2          CSL    W             SHIFT LARGER LEFT
 296 0C3DC CC           A=A-1  A             DEC EXP OF LARGER
 297 0C3DE E8   ADD40   B=B-A  A
 298 0C3E0 F9           B=-B   A             B(X)=NO OF SHIFTS TO LINE UP DP
 299 0C3E2 CD   ADD50   B=B-1  A             DEC SHIFT COUNT. BORROW?
 300 0C3E4 4A0          GOC    ADD60         YES. LINED UP
 301 0C3E7 BF7          DSR    W             SHIFT MAN OF SMALLER ARG RIGHT
 302 0C3EA 97F          ?D#0   W             SHIFTED CLEAR OFF
 303 0C3ED 5F           GOYES  ADD50         NO
 304 0C3EF 940   ADD60  ?A=B   S             LIKE SIGNS?
 305 0C3F2 52           GOYES  ADD80         YES
 306 0C3F4 832          ?SB=0                STICKY BIT SET?
 307 0C3F7 50           GOYES  ADD70         NO
 308 0C3F9 B77          D=D+1  W             YES. INCREASE D BY 1 COUNT
 309 0C3FC B7B   ADD70  C=C-D  W             SUBTRACT
 310 0C3FF 97E          ?C#0   W             MANT#0?
 311 0C402 81           GOYES  ADD90
 312                                         (+/-)0 RESULT
 313 0C404 AC0          A=0    S             FORCE SIGN POS
 314 0C407 86A          ?ST=0  sINFRD        DON'T RND TO +/-INF
 315 0C40A 01           GOYES  ADD90
 316 0C40C 86B          ?ST=0  sNEGRD        DON'T RND NEGATIVELY
 317 0C40F B0           GOYES  ADD90
 318
 319 0C411 BCC          A=-A-1 S             RND TO -INF
 320 0C414 550          GONC   ADD90         "GOTO" (A=-A-1 ALWAYS CLS CRY.)
 321
 322 0C417 A7B   ADD80  C=C+D  W             DO ADDITION
 323 0C41A A0C   ADD90  A=A-1  P             RESTORE EXP
 324 0C41D A0C          A=A-1  P
 325 0C420 A0C          A=A-1  P
 326 0C423 A0C          A=A-1  P
 327 0C426 A0C          A=A-1  P
 328 0C429 2E           P=     14
 329 0C42B AF5          B=C    W             B=MANTISSA OF ANSWER
 330 0C42E 6D40         GOTO   MPY150        normalize
 331            ****************************************************************
 332            ****************************************************************
 333            **
 334            ** Name:(S) MP2-15  -  Multiply
 335            ** Name:(S) MP15S   -  Multiply without clearing SB
 336            ** Name:(S) MULTF   -  Multiply for finite args only
 337            ** Name:(S) MP1-12  -  Multiply for one 12-form
 338            ** Name:(S) MP2-12  -  Multiply for two 12-forms
 339            **
 340            **
 341            ** Category:   MATH
 342            **
 343            ** Purpose:
 344            **      To compute  x*y
 345            **
 346            ** Entry:
 347            **      Standard floating point math input.
```

```
348            **      MULTF & MP15S: SB & XM are not cleared on entry.
349            **
350            **      CODE:  =MP2-12  GOSUB  SPLITA
351            **      ----   =MP1-12  GOSUB  SPLITC
352            **             =MP2-15  SB=0
353            **                      XM=0
354            **             =MP15S
355            **
356            **
357            **
358            ** Exit:
359            **      Standard floating point math output.
360            **      XM=1 implies 0*Inf
361            **
362            ** Calls:       (none)
363            **
364            ** Uses.......
365            **  Inclusive: P; A,B,C,D;
366            **             HD.ST.[SB,XM];
367            **
368            ** Stk lvls:   0
369            **
370            ** NOTE:
371            **      Reg. D has the 16 digit mant. of x*y if D(S)#0,
372            **        (mant of Inf & NaN is not put into D,but D(S)=0 here)
373            **      Results are truncated to 15 digits.
374            **      Unfortunately SB=1 when XM=1 on exit. (This is true for
375            **       most math routines.)
376            **
377            ***************************************************************
378            ***************************************************************
379 0C432       =MP2-12
380 0C432 7982         GOSUB  SPLITA
381 0C436 7605 =MP1-12 GOSUB  SPLITC
382 0C43A      =MP2-15
383 0C43A 822          SB=0                    RESET STICKY BIT
384 0C43D 821          XM=0
385 0C440 21   =MP15S  P=     x                P='*'
386 0C442 6091         GOTO   PWEEDS           "GOSUB" (RTN TO MULTF)
387            *       .                       .
388            *       .                       .
389            *       .                       .
390 0C446      =MULTF
391 0C446 AFD          BCEX   W                (MANTS IN C&D)
392 0C449 AC2          C=0    S
393 0C44C AC3          D=0    S
394 0C44F C0           A=A+B  A                ADD EXPONENTS
395 0C451 B4C          A=B-A  S                FORM "EXCLUSIVE OR" OF SIGNS
396 0C454 550          GONC   MPY110
397 0C457 BC8          A=-A   S                A HAS SIGN AND EXP OF ANSWER
398 0C45A AF1  MPY110  B=0    W                CLEAR B FOR PRODUCT
399 0C45D 2F           P=     15               INITIALIZE POINTER
400 0C45F 0C   MPY120  P=P+1                   POINT TO NEXT DIGIT IN MULTIPLIER
401 0C461 BF5          BSR    W                SHIFT PRODUCT RIGHT FOR NEXT DECADE
402 0C464 6600         GOTO   MPY140
```

```
403 0C468 A71   MPY130 B=B+C  W              ADD MULTIPLICAND TO PRODUCT
404 0C46B A0F   MPY140 D=D-1  P              DEC MULTIPLIER DIGIT. BORROW?
405 0C46E 59F          GONC   MPY130         NO
406 0C471 88E          ?P#    14            DONE?
407 0C474 BE           GOYES  MPY120         NO
408 0C476 AF9          C=B    W
409 0C479 AF7          D=C    W             MOVE 16 DIGIT MANTISSA TO D
410 0C47C 949   MPY150 ?B=0   S             CARRY OUT?
411 0C47F 70           GOYES  SHF10         NO
412 0C481 E4           A=A+1  A             INC EXP
413 0C483 BF5          BSR    W             SHIFT MANTISSA RIGHT
414
415             ********************************************************************
416             ********************************************************************
417             **
418             ** Name:(S) SHF10   -  Shift to normalize
419             **
420             ** Category:   MTHUTL
421             **
422             ** Purpose:
423             **      Normalize 15 form in AB.
424             **
425             ** Entry:
426             **      Finite (possibly denormalized no.) in AB
427             **
428             ** Exit:
429             **      AB is normalized (clean 0s). P=C(S), C(S)=B(S), B(S)=0
430             **      Carry clear.
431             **
432             ** Calls:      None
433             **
434             ** Uses.......
435             **  Inclusive: C(S) (see exit conditions)
436             **
437             ** Stk lvls:   0
438             **
439             ********************************************************************
440             ********************************************************************
441             * Normalize result -- N.B. P:=C(S), C(S):=B(S) on exit. This
442             *                                 is for SPLITA (to save P)
443 0C486 2E   =SHF10  P=     14
444 0C488 919          ?B=0   WP             MANTISSA ZERO?
445 0C48B 01           GOYES  SHF30         YES
446 0C48D 90D   SHF20  ?B#0   P              NORMALIZED?
447 0C490 D0           GOYES  SHF40
448 0C492 CC           A=A-1  A              DEC EXP
449 0C494 B91          BSL    WP             LEFT SHIFT MANTISSA
450 0C497 65FF         GOTO   SHF20
451 0C49B       SHF30
452 0C49B D0           A=0    A
453 0C49D 20   SHF40   P=     0
454 0C49F 80FF         CPEX   15             Restore P if saved
455 0C4A3 ACD          BCEX   S              Restore C[S] "  ". set B(S)=0.
456 0C4A6 03           RTNCC
457
```

```
458
459
460
461
462            ********************************************************************
463            ********************************************************************
464            **
465            ** Name:(S) DV2-15  -  Divide
466            ** Name:(S) DIVF    -  Divide for finite args only
467            ** Name:(S) DV15S   -  Divide without clearing SB
468            ** Name:(S) DV15M   -  Divide (same as DV2-15)
469            ** Name:(S) DV2-12  -  Divide for two 12-forms
470            **
471            **
472            ** Category:   MATH
473            **
474            ** Purpose:
475            **      To compute  y/x
476            **
477            ** Entry:
478            **      Standard floating point math input.
479            **
480            **      CODE:   =DV2-12   GOSUB  SPLTAC
481            **              =DV2-15
482            **              =DV15M    XM=0
483            **                        SB=0
484            **              =DV15S
485            **
486            **
487            ** Exit:
488            **      Standard floating point math output.
489            **      XM=1 & P=3 implies c/0 where 0<|c|<Inf
490            **           & P=4    "    0/0 or Inf/Inf
491            **
492            ** Calls:      (none)
493            **
494            ** Uses.......
495            **  Inclusive: P; A,B,C,D;
496            **             MD.ST.[SB,XM];
497            **
498            ** Stk lvls:   0
499            **
500            ** NOTE:
501            **      Divides (A,B) by (C,D) .
502            **      Results are truncated to 15 digits.
503            **
504            ********************************************************************
505            ********************************************************************
506 0C4A8      =DV2-12
507 0C4A8 7884         GOSUB  SPLTAC
508 0C4AC      =DV2-15
509 0C4AC 821  =DV15M  XM=0
510 0C4AF 822          SB=0
511 0C4B2 22   =DV15S  P=     /              P='/'
512 0C4B4 6E11         GOTO   PWEEDS         "GOSUB" (RTN TO DIVF)
```

```
513            *       .                       .
514            *       .                       .
515            *       .                       .
516 0C4B8      =DIVF                           Entry for finite arg's
517                                            CK DIVIDE XCPTIONS
518 0C4B8 AFD          BCEX    W                   (MANTS IN C&D)
519 0C4BB AC2  =DIVFCD C=0     S
520 0C4BE AC3          D=0     S
521 0C4C1 97F          ?D#0    W               ?X#0
522 0C4C4 63           GOYES   DIV100
523 0C4C6 97E  =DIVO   ?C#0    W               Y#0
524 0C4C9 C0           GOYES   DZERO           C/0
525
526                                            0/0
527 0C4CB 20           P=      EFIELD
528 0C4CD 3100         LC(2)   =eZRO/0
529 0C4D1 6D81         GOTO    INVNaN
530
531 0C4D5       DZERO                          (recall: mants in C&D )
532 0C4D5 B40          A=A-B   S               SIGN OF PRODUCT
533 0C4D8 550          GONC    DZ10
534 0C4DB BC8          A=-A    S
535 0C4DE 20   =DZ10   P=      0
536 0C4E0 3100         LC(2)   =eZRDIV         /ZERO MSG CODE
537
538 0C4E4      =DZINF                          input:A(S)=sgn, C(B)=msg
539                                                  P=0 IS ASSUMED !
540
541 0C4E4 D0           A=0     A               C/0  . CREATE INF ('FOO')
542 0C4E6 AEA          A=C     B
543                                            P=0 is assumed from DZINF entry.
544 0C4E9 3200         LCHEX   FOO             FOO
          F
545 0C4EE DE           ACEX    A                 A(X)=FOO, C(A)=msg code
546 0C4F0 AF1          B=0     W
547 0C4F3 BDD          B=-B-1  M               mant=099...99000
548 0C4F6 23           P=      DZP             P='DZ'
549 0C4F8 00           RTNSXM                  XCPT JUST OCCURRED  (XM:=1)
550
551
552 0C4FA E0   =DIV100 A=A-B   A               SUBTRACT EXPONENTS
553 0C4FC B40          A=A-B   S               FORM "EXCLUSIVE OR" OF SIGNS
554 0C4FF 550          GONC    DIV15
555 0C502 BC8          A=-A    S
556 0C505 9FF  =DIV15  ?C>=D   W               DIVIDEND .GE. DIVISOR?
557 0C508 70           GOYES   DIV120          YES
558 0C50A BF2          CSL     W               NO. SHIFT DIVIDEND LEFT
559 0C50D CC           A=A-1   A               DEC EXP OF ANSWER
560 0C50F 2E   =DIV120 P=      14              INITIALIZE POINTER
561 0C511 AF1          B=0     W               CLEAR B FOR QUOTIENT
562 0C514 6600         GOTO    DIV140
563 0C518 B05   DIV130 B=B+1   P               INC DIGIT OF QUOTIENT
564 0C518 B7B   DIV140 C=C-D   W               DIVIDEND-DIVISOR. BORROW?
565 0C51E 59F          GONC    DIV130          NO
566 0C521 A7B          C=C+D   W               YES. RESTORE DIVIDEND
```

```
567 0C524 BF2          CSL    W              SHIFT DIVIDEND LEFT FOR NEXT DECADE
568 0C527 0D           P=P-1                 POINT TO NEXT DIGIT. DONE?
569 0C529 51F          GONC   DIV140         NO
570 0C52C 6990         GOTO   SQR75
571
572
573            ********************************************************************
574            ********************************************************************
575            **
576            ** Name:(S) SQR15   -  Square Root
577            ** Name:(S) SQR17   -  SQRT for finite arguments only
578            **
579            ** Category:   MATH
580            **
581            ** Purpose:
582            **      To compute  SQRT(x)
583            **
584            ** Entry:
585            **      Standard floating point math input.
586            **
587            ** Exit:
588            **      Standard floating point math output.
589            **      XM=1 implies SQR(neg)
590            **
591            ** Calls:      (none)
592            **
593            ** Uses.......
594            **  Inclusive: P; A,B,C;
595            **             HD.ST.[SB,XM];
596            **
597            ** Stk lvls:   0
598            **
599            ** NOTE:
600            **      Certain 15-form inputs can exit with SB=0, even though
601            **      the result is inexact! e.g. SQR(1E14+1)-->1E7 & SB=0.
602            **      This occurs from BSR instr. before SQR30.
603            **
604            ********************************************************************
605            ********************************************************************
606 0C530 7B81 =SQR12  GOSUB  SPLITA
607
608 0C534      =SQR15
609 0C534 821  =SQR15M XM=0
610 0C537 822          SB=0                  (Must always be cleared here.)
611                                          (Didn't use FNPWDS in order to
612                                            keep 0 subr levels.)
613 0C53A 04           SETHEX
614 0C53C B24          A=A+1  XS
615 0C53F A2C          A=A-1  XS
616 0C542 05           SETDEC
617 0C544 5E0          GONC   SQR17          finite argument.
618
619 0C547      SQRXCP                        HANDLE EXCEPTIONAL OPERAND
620 0C547 96C          ?A#0   B              NAN?
621 0C54A 00           RTNYES
```

```
622
623 0C54C 94C          ?A#0   S            -INF?
624 0C54F 41           GOYES  SQR-
625            *   SQR(inf)   *
626
627 0C551 01           RTN                 RTN with inf
628
629
630 0C553 AC1  =SQR17  B=0    S
631 0C556 AF9          C=B    W            C=MANTISSA
632 0C559 979          ?B=0   W             ZERO INPUT?
633 0C55C 76           GOYES  SQR70        YES
634
635
636 0C55E 948          ?A=0   S             X>0?
637 0C561 C0           GOYES  SQR18
638 0C563       SQR-                        X<0.
639 0C563 20           P=     EFIELD
640 0C565 3100         LC(2)  =eSQR-       ERR CODE SQR(NEG)
641 0C569 65F0         GOTO   INVNaN
642
643 0C56D A75  SQR18   B=B+B  W
644 0C570 A75          B=B+B  W
645 0C573 A71          B=B+C  W             B=.5(MANTISSA)
646
647 0C576 D6           C=A    A
648 0C578 C6           C=C+C  A             2*EX
649 0C57A AD2          C=0    M
650 0C57D 550          GONC   SQR20        EX>=0
651                                        EX<0
652 0C580 BDE          C=-C-1 M
653 0C583       SQR20
654 0C583 D6           C=A    A
655 0C585 AFA          A=C    W
656 0C588 A76          C=C+C  W             2*EX
657 0C58B A76          C=C+C  W             4*EX
658
659 0C58E A7A          A=A+C  W             5*EXPON
660 0C591 20           P=     0
661 0C593 90C          ?A#0   P             EXP IS ODD?
662 0C596 50           GOYES  SQR30        YES
663 0C598 BF5          BSR    W             NO.SHIFT MANTISSA OF ARG RIGHT
664 0C59B BF4  SQR30   ASR    W             .5*EXPON
665 0C59E AF2          C=0    W
666 0C5A1 2E           P=     14
667 0C5A3 305          LCHEX  5
668 0C5A6 AFD          BCEX   W             C=.5 (MANTISSA)
669 0C5A9 B95  SQR35   BSR    WP            B=05000...
670 0C5AC A0D          B=B-1  P
671
672 0C5AF B05  SQR40   B=B+1  P             INCREMENT DIGIT IN ANSWER
673 0C5B2 B79  SQR50   C=C-B  W             SUBTRACT. BORROW?
674 0C5B5 59F          GONC   SQR40        NO
675 0C5B8 A79          C=C+B  W             YES. RESTORE REMAINDER
676 0C5BB BF2          CSL    W             SHIFT REMAINDER FOR NEXT DECADE
```

```
 677 0C5BE 0D          P=P-1                 POINT TO NEXT DIGIT. DONE?
 678 0C5C0 58E         GONC   SQR35         NO
 679
 680            ****************************************************************
 681            ****************************************************************
 682            **
 683            ** Name:(S) SQR70   -  Set SB according to Reg C
 684            **
 685            ** Category:   MTHUTL
 686            **
 687            ** Purpose:
 688            **      To set or clear Sticky Bit (SB) for C#0 or C=0 resp.
 689            **
 690            ** Entry:
 691            **      C=0 if SB=1 is desired, else C#0
 692            **
 693            ** Exit:
 694            **      SB=0 if C=0, else SB=1.
 695            **      Carry Clear.
 696            **
 697            ** Calls:      (none)
 698            **
 699            ** Uses.......
 700            **  Inclusive: C(A)
 701            **
 702            ** Stk lvls:   0
 703            **
 704            ** Algorithm:
 705            **      =SQR70   SB=0
 706            **               ?C=0    W
 707            **               GOYES   SQR80
 708            **               C=C-1   X
 709            **               CSR     X
 710            **       SQR80   RTNCC
 711            **
 712            ****************************************************************
 713            ****************************************************************
 714 0C5C3 822  =SQR70  SB=0                  RESET STICKY BIT
 715 0C5C6 97A  SQR75   ?C=0   W              NO REMAINDER?
 716 0C5C9 80           GOYES  SQR80          TRUE
 717 0C5CB A3E          C=C-1  X              LSD=9
 718 0C5CE BB6          CSR    X              SET STICKY BIT
 719 0C5D1 03    SQR80  RTNCC                 (needed for RADIUS)
 720            * ---------------------------------------------
 721
 722
 723
 724
 725 0C5D3      PWEEDS                        PULL WEEDS (INF & NaN)
 726                                           INPUT: P= pending op. (+,*,/)
 727                                                  (A,B)=Y,(C,D)=X
 728                                                  STATUS BITS: FOR RND,INF &
 729                                                               USER MODES.
 730 0C5D3 AC1          B=0    S              Needed for 0 mant tests and to
 731 0C5D6 AC3          D=0    S              indicate no 16 digit result in
```

```
                                          D after MP2-15.
0C5D9 04          SETHEX
0C5DB B26         C=C+1  XS
0C5DE A2E         C=C-1  XS
0C5E1 4F2         GOC    X=XCP            X NOT FINITE

0C5E4 B24         A=A+1  XS
0C5E7 A2C         A=A-1  XS
0C5EA 05          SETDEC
0C5EC 460         GOC    YCP;X#
0C5EF 66DD        GOTO   X&Y#S            X&Y ARE FINITE NUMBERS

0C5F3     YCP;X#                        Y NOT FINITE, X FINITE
0C5F3 96C         ?A#0   B              Y=NAN?  (X=#)
0C5F6 00          RTNYES

          *    Y=INF & X=#
0C5F8 892            ?P=    /
0C5FB 33             GOYES  SGNOF*      INF/#
0C5FD 890            ?P=    +
0C600 00             RTNYES

                                          P=*
0C602 97F         ?D#0   W              X#0? (Y=INF)
0C605 92          GOYES  SGNOF*
                                          X=0

          ********************************************************************
          ********************************************************************
          **
          ** Name:(S) INF*0   -  Inf*0 exception
          **
          ** Category:   MTHUTL
          **
          ** Purpose:
          **      To create a 15-form NaN result with Inf*0 msg code.
          **
          ** Entry:
          **      No conditions.
          **
          **      CODE:  =INF*0  P=     0
          **                     LC(2)  =eIF*ZR
          **                     GOTO   INVNaN
          **
          **
          **
          ** Exit:
          **      (See INVNaN)
          **
          ** Calls:      Goes to INVNaN
          **
          ** Uses.......
          **  Inclusive: P; A,B,C(A); HD.ST.[XM,SB]
          **
          ** Stk lvls:   0
```

```
787             **
788             ********************************************************************
789             ********************************************************************
790 0C607        =INF*0                          0*INF
791 0C607 20            P=      EFIELD
792 0C609 3100          LC(2)   =eIF*ZR         MSG CODE
793 0C60D 6150          GOTO    INVNaN
794
795
796
797
798
799
800 0C611        X=XCP                           X IS NOT FINITE
801 0C611 05            SETDEC
802 0C613 922           ?A=C    XS              Y EXCPTN TOO?
803 0C616 F2            GOYES   Y&XXCP
804                                             Y=#
805 0C618 96E           ?C#0    B               X=NAN (&Y=#)
806 0C61B E1            GOYES   xyex           Y:=X=NAN
807
808             *     Y=# & X=INF
809 0C61D 892           ?P=     /
810 0C620 D1            GOYES   DIVOP
811 0C622 890           ?P=     +
812 0C625 41            GOYES   xyex            INF RESULT
813                                             P=x
814 0C627 979           ?B=0    W               Y=0? (X=INF)
815 0C62A DD            GOYES   INF*0
816 0C62C DA            A=C     A               INF EXPON
817
818 0C62E        SGNOF*                          SIGN OF PRODUCT
819 0C62E B4A           A=A-C   S
820 0C631 500           RTNNC
821 0C634 BC8           A=-A    S
822 0C637 01            RTN
823
824
825 0C639 6D50   xyex    GOTO    XYEX
826
827
828 0C63D D0     DIVOP   A=0     A               (CARRY MUST BE SET ON ENTRY)
829 0C63F AF1           B=0     W
830 0C642 4BE           GOC     SGNOF*          "GOTO"
831
832 0C645        Y&XXCP
833 0C645 96E           ?C#0    B               X=NAN?
834 0C648 A4            GOYES   XNNYCP
835                                             X=INF Y=XCPT
836 0C64A 96C           ?A#0    B               Y=NAN?
837 0C64D 00            RTNYES                  Y=NAN,X=INF
838
839             *     X&Y INF
840 0C64F 890           ?P=     +
841 0C652 33            GOYES   INFS+
```

```
842 0C654 891          ?P=    x
843 0C657 7D           GOYES  SGNOF*
844
845            *    INF/INF
846 0C659 20           P=     EFIELD
847 0C65B 3100         LC(2)  =eIF/IF
848
849
850
851
852            ****************************************************************
853            ****************************************************************
854            **
855            ** Name:(S) INVNaN  -  Create IVL NaN
856            **
857            ** Category:   MATH
858            **
859            ** Purpose:
860            **      To create an internal NaN and set XM for IVL.
861            **
862            ** Entry:
863            **      C(B)=two nib. mainframe error msg code.
864            **
865            ** Exit:
866            **      (A,B):=NaN with B(14..11):= 4nib msg code
867            **      C(A):= 4nib msg code for input to MESSAGE ROUTINE.
868            **      XM:=1 (indicates xcpt'n) & P:=IVP (IV xcpt'n)
869            **      B(XS)=9 (if in DEC MODE !). This indicates a
870            **       15-form INVNaN (i.e. created in math routine -- input
871            **       NaNs from SPLITA will have F instead of 9 in B(XS)).
872            **       This causes INVNaNs (and their encoded message)
873            **       to be more significant than input NaNs and thus
874            **       will be preserved when two NaNs enter a function.
875            **
876            **      i.e.  A = 0000000000000F01
877            **            B = 000mm00000000900
878            **            C = ----------000mm
879            **
880            **
881            ** Calls:      (none)
882            **
883            ** Uses.......
884            **  Inclusive: P; A,B,C(A);
885            **             HD.ST.[XM,SB];
886            **
887            ** Stk lvls:   0
888            **
889            ** NOTE:
890            **       CAUTION: This routine will set SB (unfortunately).
891            **
892            ****************************************************************
893            ****************************************************************
894 0C65F      =INVNaN                          entry for 2 nib err code.
895 0C65F AF1          B=0    W
896 0C662 AE5          B=C    B                 B(A)=msg code (in mant w/ lead 0s)
```

```
897
898 0C665 AF0          A=0    W
899 0C668 20           P=     0
900 0C66A 3210         LCHEX  F01
          F
901 0C66F ABA          A=C    X              A(X):=F01  (NaN expon)
902 0C672 D9           C=B    A              C(A)=code for err msg
903
904 0C674 0C    IV50   P=P+1
905 0C676 815          BSRC                  shift 4 digit msg number (5 times)
906 0C679 885          ?P#    5
907 0C67C 8F           GOYES  IV50
908 0C67E BAD          B=-B-1 XS             B(XS):= 9 (in DEC MODE ! )
909
910 0C681 24           P=     IVP
911 0C683 00   SETXM   RTNSXM                XCPT JUST OCCURRED (XM:=1)
912
913
914
915
916
917 0C685       INFS+
918 0C685 20           P=     EFIELD
919 0C687 3100         LC(2)  =eIF-IF         ERR CODE INF-INF
920 0C68B 946          ?A#C   S              DIFF. SIGNS
921 0C68E 1D           GOYES  INVNaN
922                                          SAME SIGNS
923 0C690 01           RTN
924
925 0C692       XNNYCP                       X=NAN,Y=XCPT
926 0C692 96C          ?A#0   B              Y=NAN?
927 0C695 01           GOYES  TWONaN
928
929                                          Y=INF, X=NAN
930
931
932             ********************************************************************
933             ********************************************************************
934             **
935             ** Name:(S) XYEX    -  EXCHANGE X & Y
936             **
937             ** Category:   MTHUTL
938             **
939             ** Purpose:
940             **      To exchange the internal nos. Y=(A,B) & X=(C,D) .
941             **
942             ** Entry:
943             **      (A,B)=Y & (C,D)=X
944             **
945             ** Exit:
946             **      (A,B)=X & (C,D)=Y
947             **      Does not alter carry
948             **
949             ** Calls:      (none)
950             **
```

```
                ** Uses.......
                **  Inclusive: A,B,C,D
                **
                ** Stk lvls:   0
                **
                ** Detail:
                **      Swaps entire regs (A with C and B with D)
                **
                ****************************************************************
                ****************************************************************
0C697           =XYEX                           EXCHANGE (A,B)=Y WITH (C,D)=X
0C697 AFD               BCEX    W
0C69A AFF               CDEX    W
0C69D AFD               BCEX    W
0C6A0 AFE               ACEX    W
0C6A3 01                RTN                     must not alter carry.



0C6A5           =TWONaN                        X=Y=NAN. Return most significant NaN
                                                 [i.e. smallest (SIG#, ERR#) ]

0C6A5 AFF               CDEX    W               for compare
0C6A8 9A1               ?C<B    XS              X<<Y ?  (significance digit)
0C6AB C0                GOYES   TWON1           set carry.
0C6AD 9A5               ?C>B    XS
0C6B0 00                RTNYES
0C6B2 9D1               ?C<B    M               X<Y
0C6B5 20                GOYES   TWON1           set carry.
0C6B7 AFF    TWON1      CDEX    W
0C6BA 4CD               GOC     XYEX            return most significant NaN
0C6BD 01                RTN


                ************************************************



                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) SPLITA  -  SPLIT A
                **
                ** Category:   MTHUTL
                **
                ** Purpose:
                **      To convert an external (12 dig.) form into an internal
                **      (15 dig.) form
                **
                ** Entry:
                **      A=x' (external no.)
                **
                ** Exit:
                **      (A,B)=x (normal internal form of x')
```

```
1006            **      CR.set => excptn'l operand (i.e. NaN or Inf )
1007            **      CR.clr => finite operand
1008            **      --see DETAIL below for normal internal form defn.
1009            **
1010            ** Calls:      (none)
1011            **
1012            ** Uses.......
1013            **  Inclusive: A,B   (actually B[14..5] ends up in A[14..5] )
1014            **
1015            ** Stk lvls:   0
1016            **
1017            ** Detail:
1018            **      DEFN: The "normal internal form" of
1019            **              1) NaN is A(A)=00F01 & B(XS)=F (i.e.mant#0).
1020            **              2) Inf is A(A)=00F00 &  "    .
1021            **              3) finite no is a normalized no.(no denorm.).
1022            **
1023            ****************************************************************
1024            ****************************************************************
1025 0C6BF ADC  =SPLITA ABEX   M
1026 0C6C2 AB8          B=A    X
1027 0C6C5 A24          A=A+A  XS              NEG 3 DIGIT EXP?
1028 0C6C8 D0           A=0    A
1029 0C6CA 540          GONC   SPLA10          NO
1030 0C6CD CC           A=A-1  A               5 9'S
1031 0C6CF AB4  SPLA10  A=B    X
1032 0C6D2 AB1          B=0    X
1033 0C6D5      =NRMLAB
1034 0C6D5 04           SETHEX
1035 0C6D7 B24          A=A+1  XS
1036 0C6DA A2C          A=A-1  XS
1037 0C6DD 05           SETDEC
1038 0C6DF 4C0          GOC    NMNF
1039 0C6E2 AC5          B=C    S               Save C[S] in B[S]
1040 0C6E5 80FF         CPEX   15              Save P in C[S]
1041 0C6E9 5B4          GONC   Shf             "GOTO SHF10"
1042
1043 0C6EC AD0   NMNF   A=0    M               00Fxx for expon
1044 0C6EF AA8          B=A    XS              make mant non-zero
1045
1046 0C6F2 02           RTNSC                  carry set on excpt'l operand.
1047
1048
1049
1050
1051
1052
1053            ****************************************************************
1054            ****************************************************************
1055            **
1056            ** Name:(S) CLRFRC  -  Clear fractional part
1057            **
1058            ** Category:   MTHUTL
1059            **
1060            ** Purpose:
```

```
1061            **      Clears fractional part of quantity in A/B, preserving
1062            **      the sign of the argument.  Returns the result in A/B.
1063            **      Carry set if no fractional part.
1064            **
1065            ** Entry:
1066            **      A/B -- 15-digit form of quantity
1067            **
1068            ** Exit:
1069            **      A/B -- quantity with fractional part cleared
1070            **      Carry=set if no fractional part
1071            **      Carry=clear otherwise
1072            **      DECMODE
1073            **
1074            ** Calls:       INFR15
1075            **
1076            ** Uses.......
1077            **  Inclusive: A(A),B,C(A),P
1078            **
1079            ** Stk lvls:   2
1080            **
1081            ** History:
1082            **
1083            **    Date      Programmer              Modification
1084            **  --------    ----------   ---------------------------------
1085            **  08/24/82    PM           Documented routine
1086            **  09/23/82    SB           Packed out =CLFRCF Entry
1087            **  12/02/82    JT           Corrected docum. for stk lvls.
1088            **                            (INFR15 calls FINITA now)
1089            ******************************************************************
1090            ******************************************************************
1091 0C6F4 7540 =CLRFRC GOSUB  INFR15         locate decimal point
1092 0C6F8 89F          ?P=    15             integer or exceptional number?
1093 0C6FB 00           RTNYES
1094 0C6FD 919          ?B=0   WP             all integer?
1095 0C700 00           RTNYES
1096 0C702 A91          B=0    WP             clear fractional part,preserve sign
1097 0C705 88E          ?P#    14
1098 0C708 40           GOYES  CLREND
1099 0C70A D0           A=0    A              all fractional: clear expon too.
1100 0C70C 03   CLREND  RTNCC
1101            ******************************************************************
1102            ******************************************************************
1103            **
1104            ** Name:    FRAC15  -  Compute Fractional Part of Argument
1105            **
1106            ** Category:   MTHUTL
1107            **
1108            ** Purpose:
1109            **      Compute fractional part of argument, preserving sign.
1110            **
1111            ** Entry:
1112            **      Argument in A/B.
1113            **
1114            ** Exit:
1115            **      Result in A/B.
```

```
1116            **
1117            ** Calls:      FINITA, INFR15, SHF10 (falls through)
1118            **
1119            ** Uses.......
1120            **             A[A],B,C[S],C[A],P.
1121            **
1122            ** Stk lvls:   2
1123            **
1124            ****************************************************************
1125            ****************************************************************
1126 0C70E 71F5 =FRAC15 GOSUB  FINITA         test for nonfinite
1127 0C712 5E0          GONC   FRC15F
1128 0C715 96C          ?A#0   B              NaN?
1129 0C718 00           RTNYES
1130 0C71A D0    A/B=0  A=0    A              +-inf: return +-0.
1131 0C71C AF1          B=0    W
1132 0C71F 01           RTN
1133
1134 0C721 7810 =FRC15F GOSUB  INFR15         finite
1135 0C725 0C           P=P+1
1136 0C727 0C    FRCITR P=P+1
1137 0C729 4B0          GOC    ENDINT
1138 0C72C BF1          BSL    W
1139 0C72F CC           A=A-1  A
1140 0C731 65FF         GOTO   FRCITR
1141 0C735      ENDINT
1142 0C735 605D Shf     GOTO   SHF10            ( B(S):=0 in SHF10 )
1143
1144
1145            ****************************************************************
1146            ****************************************************************
1147            **
1148            ** Name:(S) IF12A   -  Integer/Fraction Split
1149            ** Name:(S) INFR15  -  Integer/Fraction Split
1150            **
1151            **  Category:  MTHUTL
1152            **
1153            **  Purpose:  Find decimal (used by INT15 & FRAC15). Returns
1154            **            position of decimal encoded in P (see below).
1155            **
1156            **  Entry: Standard Math - 12 dig: IF12A,  15 dig: INFR15
1157            **
1158            **  Exit:  Encoded location of decimal in P.
1159            **
1160            **  Alters:
1161            **        IF12A:  A,B,C[A],P,CARRY
1162            **        INFR15: C[A],P,CARRY
1163            **
1164            **  Stk Lvls: 1
1165            **
1166            **  Note:
1167            **             ARGUMENT      RETURN (P)   [Notation: EXP(X)=E]
1168            **           ------------    ----------
1169            **            NaN or INF         15
1170            **            Standard 0         13 (standard 0 has E=0)
```

```
1171                **
1172                **          E<0              14
1173                **          0<=E<=13         13-E
1174                **          13<E             15
1175                **
1176                **
1177                **  Note:  If the Expon=14 (i.e. a 15 digit integer) then C(A)
1178                **         is 0. If Expon>14 (but finite) then C(A)=50000 on exit
1179                **         This is used in YX15 to determine if x is an even
1180                **         integer.
1181                **
1182                **
1183                **  History:
1184                **
1185                **     Date      Programmer            Modification
1186                **    --------   ----------   ---------------------------------
1187                **    09/23/82   SB           15-dig entry: P=15 for NaN or INF,
1188                **                            Comments, description update,
1189                **                            Standard header.
1190                **
1191                ********************************************************************
1192                ********************************************************************
1193 0C739 728F =IF12A  GOSUB  SPLITA
1194 0C73D 72C5 =INFR15 GOSUB  FINITA
1195 0C741 443         GOC    BIG15
1196 0C744 20          P=     0
1197 0C746 D2          C=0    A
1198 0C748 3131        LCHEX  13              C(A)=00013
1199 0C74C E2          C=C-A  A
1200 0C74E 441         GOC    BIGSML          If EXP<0 or EXP>13, Go BIGSML
1201 0C751 21   =DECP=C P=     1
1202 0C753 90A         ?C=0   P
1203 0C756 70          GOYES  PICKUP          If EXP<10, Go PICKUP
1204 0C758 29          P=     9                 Else create Pickup value (HEX)
1205 0C75A 809         C+P+1                    by adding 10 to C[0].
1206 0C75D 80D0  PICKUP P=C    0
1207 0C761 01          RTN
1208
1209 0C763 E6    BIGSML C=C+1  A
1210 0C765 401         GOC    BIG15           expon = 14 (rtn with C(A)=0)
1211 0C768 3400        LCHEX  50000             else C(A)=50000.
           005
1212 0C76F 2E          P=     14
1213 0C771 8BE         ?A>=C  A
1214 0C774 00          RTNYES
1215 0C776 2F    BIG15 P=     15
1216 0C778 01          RTN
1217                *-------------------------------------------------------------*
1218
1219
1220
1221
1222
1223
1224
```

```
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236            *------------------------------------------------------*
1237            ********************************************************
1238            *                                                      *
1239            *        REMAINDER FCNS. --  RMD,MOD & REM             *
1240            *                                                      *
1241            ********************************************************
1242            *------------------------------------------------------*
1243
1244
1245
1246
1247
1248
1249
1250            ************************************************************************
1251            ************************************************************************
1252            **
1253            ** Name:    RMD15   -  RMD function
1254            **
1255            ** Category:   MATH
1256            **
1257            ** Purpose:
1258            **      To compute  xRMDy = x-y*IP(x/y)
1259            **
1260            ** Entry:
1261            **      Standard floating point math input.
1262            **
1263            ** Exit:
1264            **      Standard floating point math output.
1265            **
1266            ** Uses.......
1267            **  Inclusive: P; A,B,C,D; R regs [0,1]; ST.[NOTMOD (s8)];
1268            **             HD.ST.[SB,XM];
1269            **
1270            ** Stk lvls:   2
1271            **
1272            ** NOTE:
1273            **      RMD is the BASIC fcn REM. It gives exact results every-
1274            **      where but is not periodic in x across x=0.
1275            **
1276            ************************************************************************
1277            ************************************************************************
1278
1279 0C77A 76B1 =RMD12  GOSUB  SPLTAC
```

```
1280 0C77E      =RMD15                          xRMDy=x-y*IP(x/y)
1281 0C77E 858          ST=1   NOTMOD
1282 0C781 7231         GOSUB  REMPWD           need out xcpt'l args
1283
1284 0C785 AC8          B=A    S                Sg(R):=Sg(x)
1285 0C788 76C0         GOSUB  REDUCe
1286 0C78C 400          RTNC                    RTN WITH X   (|x|<|y|)
1287 0C78F 5F5          GONC   RSTEXP           "goto" (rtn: Sg(x)*R )
1288            *   ----------------------------------------
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300            ************************************************************
1301            ************************************************************
1302            **
1303            ** Name:    MOD15   -  MOD function
1304            **
1305            ** Category:   MATH
1306            **
1307            ** Purpose:
1308            **      To compute  xMODy = x-y*INT(x/y)
1309            **
1310            ** Entry:
1311            **      Standard floating point math input.
1312            **
1313            **
1314            ** Exit:
1315            **      Standard floating point math output.
1316            **
1317            ** Uses.......
1318            **  Inclusive: P; A,B,C,D; R regs [0,1]; ST.[s=SGNS(s9),
1319            **             NOTMOD(s8)]; HD.ST.[SB,XM];
1320            **
1321            ** Stk lvls:   2
1322            **
1323            ** NOTE:
1324            **      MOD is subject to round'g error unlike RMD, but MOD is
1325            **      periodic in x over the entire x-axis.
1326            **
1327            ************************************************************
1328            ************************************************************
1329 0C792    =MOD12
1330 0C792 7E91         GOSUB  SPLTAC
1331 0C796    =MOD15                          xMODy=x-y*INT(x/y)
1332 0C796 848          ST=0   NOTMOD          FOR Y=inf ONLY (in REMPWD)
1333 0C799 72A0         GOSUB  SGNS            to set s=SGNS approp'ly
1334 0C79D 7611         GOSUB  REMPWD          need out xcpt'l args
```

```
1335 0C7A1 AC5          B=C    S              Sg(R):=Sg(y)
1336 0C7A4 7AA0         GOSUB  REDUCe
1337 0C7A8 421          GOC    MODX<Y
1338
1339            *   usual x>=y case   *
1340 0C7AB 879          ?ST=1  s=SGNS         SAME SIGNS ?
1341 0C7AE 14           GOYES  RSTEXP          (Rtn with Sg(y)*R )
1342
1343 0C7B0 979          ?B=0   W              R'=0?  (i.e. x-ky=0 the discont'ity
1344 0C7B3 C3           GOYES  RSTEXP         USE SG(Y) FOR K=0
1345 0C7B5 B7D          B=C-B  W              R':=Y'-R' [i.e.-(!y!-!R!)x10^ex(y)]
1346 0C7B8 563          GONC   RSTEXP         "GOTO"
1347
1348
1349 0C7BB      MODX<Y                        !X!<!Y! case
1350
1351            *   Bring back y   *
1352 0C7BB AF7          D=C    W
1353 0C7BE 118          C=R0                  (C,D)= y
1354
1355            *   Watch out for MOD(0,y)   *
1356 0C7C1 879  MOD50   ?ST=1  s=SGNS         x & y have same signs ?
1357 0C7C4 00           RTNYES                RTN X
1358
1359            *     Test M-field since x is finite at this point but     *
1360            *     B[S] may not be clean when coming from REMPWDS !     *
1361 0C7C6 959          ?B=0   M              X=0?
1362 0C7C9 00           RTNYES                RTN +/- 0
1363
1364                                          Since !x!<!y! => (x+y)#0, ignore
1365                                          rnd'g bits input.(=> won't use s11)
1366 0C7CB 6A9B         GOTO   AD15M           (want SB=1 if add is inex. else
1367                                           SB=0)
1368                                          x+y  (=Sg(y)*(|y|-|x|))
1369           *   ----------------------------------------------
1370
1371
1372
1373
1374
1375           ********************************************************************
1376           ********************************************************************
1377           **
1378           ** Name:    REM15   -  IEEE "REM" function
1379           **
1380           ** Category:   MATH
1381           **
1382           ** Purpose:
1383           **      To compute  xREMy = x-y*NI(x/y)  where NI(x)=nearest int.
1384           **
1385           ** Entry:
1386           **      Standard floating point math input.
1387           **
1388           ** Exit:
1389           **      Standard floating point math output.
```

```
1390            **
1391            ** Uses.......
1392            **  Inclusive: P; A,B,C,D; R regs [0,1]; ST.[NOTMOD(s8)];
1393            **             HD.ST.[SB,XM];
1394            **
1395            ** Stk lvls:   2
1396            **
1397            ** NOTE:
1398            **      REM is defined by the IEEE Floating Point Standard. It
1399            **      gives exact results and is periodic in x almost everywher
1400            **      with per=y. Only at multiples of y/2 does the period
1401            **      double.
1402            **
1403            ********************************************************************
1404            ********************************************************************
1405 0C7CF 7161 =REM12  GOSUB  SPLTAC
1406 0C7D3      =REM15                          xREMy=x-y*NI(x/y)
1407 0C7D3 858          ST=1   NOTMOD
1408 0C7D6 7DD0         GOSUB  REMPWD           weed out excpt'l args
1409 0C7DA AC8          B=A    S                Sg(R):=Sg(x)
1410 0C7DD 7170         GOSUB  REDUCe
1411 0C7E1 473          GOC    REMX<Y
1412
1413            *  usual |x|>=|y| case. (Here R & y have same expon, so use mants
1414            *                        2R' & y' for testing R<y/2, >y/2 & =y/2.
1415            *
1416            *                                                          *
1417            *    B=mant(R), C=mant(y)                                  *
1418            *    R0=Sg(x)..ex(y)=ex(R), D(XS)=last digit of quot.      *
1419            *                                                          *
1420
1421
1422
1423 0C7E4 AF4  R?Y/2   A=B    W
1424 0C7E7 A74          A=A+A  W               2R'
1425
1426 0C7EA 9FE          ?A>=C  W               2!R! >= !Y!?
1427 0C7ED 90           GOYES  REM120
1428
1429            * R < Y/2  *
1430 0C7EF 110  RSTEXP  A=R0
1431 0C7F2 639C  shf10  GOTO   SHF10            NORML'ZE R
1432
1433
1434 0C7F6 972  REM120  ?A=C   W               !R!=!Y!/2?
1435 0C7F9 C0           GOYES  R=Y/2
1436 0C7FB 110  R>Y/2   A=R0
1437 0C7FE B7D          B=C-B  W               (!y!-!R!)x10^ex(y)
1438 0C801 6110         GOTO NEGT              RTN: -Sg(x)*(!y!-!R!)*10^ex(y)
1439
1440 0C805 110  R=Y/2   A=R0                   restore S(R) & ex(y)
1441
1442 0C808 AE3          D=0    B
1443 0C80B 81F          DSRB                   Test D(XS) even or odd.
1444 0C80E 96B          ?D=0   B               Is last quot digit even ?
```

```
1445 0C811 1E          GOYES  shf10
1446 0C813 BCC  NEGT   A=-A-1 S            -Sg(x)*!R!
1447 0C816 5BD         GONC   shf10        "GOTO"
1448
1449
1450
1451
1452
1453           *   |x| < |y|   case   *
1454
1455 0C819      REMX<Y
1456 0C819 AA3         D=0    XS           force even quot. since DIV=0 for
1457                                        !X!<!Y!
1458 0C81C 128         CROEX
1459 0C81F EA          A=A-C  A            EX(X)-EX(Y)
1460 0C821 128         CROEX
1461 0C824 8A8         ?A=0   A            ex(x)=ex(y) ?
1462 0C827 DB          GOYES  R?Y/2         (Use R*10^ex(y) = x)
1463
1464           *   ex(x) # ex(y)  *
1465 0C829 E4  REM310  A=A+1  A            ex(x)-ex(y)+1
1466 0C82B 470         GOC    REM330       ex(x)=ex(y)-1
1467                                        ex(x)<=ex(y)-2  ( !x!<!y!/2 )
1468
1469           *   Definitely |x| < |y|/2   *
1470 0C82E 111 RTNX    A=R1                 Sg(x)..ex(x)
1471 0C831 01          RTN                  RTN x
1472
1473           *  ex(x) = ex(y) - 1   *
1474           *          Adjust mant(y) so that R=x & y have common exponent
1475           *          ex(y)-1. (NOTE: If ex(y) were used with a BSR on 2x,
1476           *           a digit could be lost with 15 form inputs causing
1477           *           the test R>Y/2 to fail.)
1478
1479
1480 0C833      REM330
1481 0C833 BF2         CSL    W            10*mant(y)
1482 0C836 111         A=R1
1483 0C839 100         R0=A                R0=Sg(x)..ex(y) - 1  here.
1484 0C83C 47A         GOC    R?Y/2         "GOTO",
1485                                        Use (x' & 10y')*10^ex(x) = x & y
1486
1487           *   ------------------------------------------
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
```

```
1500
1501            ***********************************************
1502            *  Support routines for Remainder Functions   *
1503            ***********************************************
1504
1505
1506
1507
1508
1509            * ------------------------------------------------------ *
1510            *  SGNS - Signs                                          *
1511            *               INPUT:   x & y (15 forms)                *
1512            *               OUTPUT:  B(S)=Sign of x*y  &  s=SGNS is set  *
1513            *                         iff Sg(x)=Sg(y).               *
1514            *                                                        *
1515
1516
1517 0C83F    SGNS                            SGN OF A PRODUCT
1518 0C83F AC1         B=0    S
1519 0C842 859         ST=1   s=SGNS
1520 0C845 942         ?A=C   S
1521 0C848 00          RTNYES                 B(S)="+" & =SGNS is true.
1522
1523 0C84A 849         ST=0   s=SGNS          DIFF SIGNS
1524 0C84D BCD         B=-B-1 S               B(S)="-"
1525 0C850 01          RTN
1526            * ---------------------------------- *
1527
1528
1529
1530
1531
1532
1533
1534            ********************************************************************
1535            ********************************************************************
1536            **
1537            ** Name:    REDUCe  -  Reduce |x| by |y|
1538            **
1539            ** Category:   MATH
1540            **
1541            ** Purpose:
1542            **      To reduce |x| by |y| for finite arguments. [RMD(|x|,|y|)]
1543            **      This is the heart of the RMD,MOD and RED functions.
1544            **
1545            ** Entry:
1546            **      (A,B) = x, (C,D) = y, and B(S)=Sg(R).
1547            **
1548            ** Exit:  (where R denotes the remainder of the reduction)
1549            **
1550            **   1) If |x|<|y| then Carry Set and (A,B)=x,
1551            **      C=mant(y), R0=Sg(R)..ex(y) and R1=Sg(x)..ex(x)
1552            **
1553            **   2) If |x|>=|y| then Carry Clear and B=mant(R),
1554            **      C=mant(y), R0=Sg(R)..ex(y), D(XS)=last digit of
```

```
1555            **      quotient.
1556            **
1557            ** Calls:      (none)
1558            **
1559            ** Uses.......
1560            **  Inclusive: A,B,C,D; R regs [0,1];
1561            **
1562            ** Stk lvls:   0
1563            **
1564            ********************************************************************
1565            ********************************************************************
1566
1567 0C852      =REDUCe
1568 0C852 101          R1=A
1569 0C855 AC9          C=B     S
1570 0C858 AC1          B=0     S
1571 0C85B AC3          D=0     S
1572 0C85E 108          R0=C
1573 0C861 979          ?B=0    W               X=0?
1574 0C864 34           GOYES   RED0<Y
1575
1576 0C866 AFD  RED50   BCEX    W
1577 0C869 E0           A=A-B   A               ex(X)-ex(Y)=#div-1
1578 0C86B 9FF          ?C>=D   W               MANT X >= MANT Y ?  (X' >= Y' ? )
1579 0C86E 70           GOYES   RED90
1580 0C870 CC           A=A-1   A               adjust #div-1 = ex(X DIV Y)
1581 0C872 BF2          CSL     W               form X'
1582
1583 0C875 D8   RED90   B=A     A
1584 0C877 C5           B=B+B   A               2(#div-1). carry if #div<=0   (X<Y)
1585 0C879 AC4          A=B     S               A=Sg(R)..#div-1
1586 0C87C AF5          B=C     W               B=mant(x')
1587 0C87F AFF          CDEX    W               C=mant(y), D=Sg(R)..ex(y)
1588 0C882 472          GOC     REDX<Y          !X! < !Y! CASE
1589 0C885 101          R1=A
1590
1591            *    Here  R1=Sg(R)..#div-1                        *
1592            *          A(A)=#div-1, B=mant(x'), C=mant(y)      *
1593
1594
1595            *                               Reduction for RMD,MOD & REM (RED)
1596            *                               combined to save code (some loss
1597            *                                is speed).
1598
1599 0C888 550          GONC    RMOD35          "GOTO"
1600
1601 0C88B BF1   RMOD30 BSL     W
1602 0C88E AA3   RMOD35 D=0     XS              init'ze quot. digit
1603
1604 0C891 B27   RMOD40 D=D+1   XS              Build quotient digit
1605 0C894 B71          B=B-C   W                and reduce |x| by |y|.
1606 0C897 59F          GONC    RMOD40
1607
1608 0C89A A71          B=B+C   W               restore remainder
1609 0C89D CC           A=A-1   A
```

```
1610 0C89F 5BE          GONC   RMOD30
1611
1612 0C8A2 A2F          D=D-1  XS              restore last quot digit
1613 0C8A5 03           RTNCC                  C.C.for normal reduction (|x|>=|y|)
1614
1615
1616
1617
1618 0C8A7 AFB  REDO<Y  C=D    W               Puts mant y into C to make use
1619                                            of REDX<Y code (Note D(P)=0 here)
1620 0C8AA      REDX<Y
1621 0C8AA 111          A=R1                   Restore Sg(x) & Expon(x)
1622 0C8AD 949          ?B=0   S
1623 0C8B0 50           GOYES  RD300
1624 0C8B2 BF5          BSR    W               mant X restored from X'
1625 0C8B5 02    RD300  RTNSC                  C.S. for |x|<|y|.
1626            * ------------------------------- *
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636            * ------------------------------------------------------ *
1637            *   REMPWD                                               *
1638            *                                                        *
1639            *     INPUT:   (A,B)=x, (C,D)=y,  NOTMOD=1 iff not MOD fcn.  *
1640            *                                                        *
1641            *     OUTPUT:                                            *
1642            *       1) If either x or y = NaN then return from calling   *
1643            *          routine with (A,B)= the correct NaN.          *
1644            *                                                        *
1645            *       2) If |x|=Inf then return from calling routine with  *
1646            *          "INVALID ARG" NaN (XM=1) .                    *
1647            *                                                        *
1648            *       3) If MOD(x,+/-Inf), then directly exit MOD through  *
1649            *          MOD50.                                        *
1650            *                                                        *
1651            *       4) If |y|=0 then "INVALID ARG" NaN (with XM=1).     *
1652            *                                                        *
1653            *       5) Else return to calling routine with same inputs.  *
1654            *                                                        *
1655            *                                                        *
1656            *     ALSO USES: R0                                      *
1657
1658
1659
1660 0C8B7      REMPWD
1661 0C8B7 821          XM=0
1662 0C8BA 822          SB=0
1663 0C8BD 8E00         GOSUBL =MSN15
           00
```

```
1664 0C8C3 560          GONC   REMP2
1665 0C8C6 07           C=RSTK
1666 0C8C8 01           RTN
1667
1668 0C8CA 7534  REMP2  GOSUB  FINITA
1669 0C8CE 5A0          GONC   REMP5          X FINITE
1670
1671            *   |x|=Inf    *
1672 0C8D1 07   GIVARG  C=RSTK                TO RTN FROM REM FCNS
1673 0C8D3 8C00         GOLONG =IVARG          "INVALID ARG" NaN
           00
1674
1675 0C8D9 7234 REMP5   GOSUB  FINITC         Y
1676 0C8DD 531          GONC   REMP20         Y FINITE
1677
1678            *   |y| = Inf   *
1679 0C8E0 108          R0=C
1680 0C8E3 07           C=RSTK                y non-finite. don't return to
1681 0C8E5 118          C=R0                      reduction code
1682
1683 0C8E8 878          ?ST=1  NOTMOD
1684 0C8EB 00           RTNYES                RTN X
1685
1686            *   MOD(x,Inf)  *
1687 0C8ED 63DE         GOTO   MOD50
1688
1689 0C8F1 95F  REMP20  ?D#0   M              Y#0? (use M-field--D[S] may not be
1690 0C8F4 00           RTNYES                 clean; ok here, since y is finite
1691 0C8F6 5AD          GONC   GIVARG
1692            * ------------------------------- *
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709            ********************************************************************
1710            ********************************************************************
1711            **
1712            ** Name:    IDIV15  -  DIV (integer division)
1713            **
1714            ** Category:   MATH
1715            **
1716            ** Purpose:
1717            **      To compute  xDIVy = IP(x/y)
```

```
1718                **
1719                ** Entry:
1720                **      Standard floating point math input.
1721                **
1722                ** Exit:
1723                **      Standard floating point math output.
1724                **
1725                ** Uses.......
1726                **  Inclusive: P; A,B,C,D;
1727                **             HD.ST.[SB,XM];
1728                **
1729                ** Stk lvls:   1
1730                **
1731                ** NOTE:
1732                **      This routine was changed to save code, and now it may
1733                **      occasionally signal inexact (SB=1) when the result is
1734                **      actually exact.
1735                **
1736                **      Signals exact (SB=0) when :
1737                **
1738                **       1) P#15. Since this indicates x/y has less than or
1739                **          equal to 14 digits in its integer part, IP(x/y)
1740                **          will be exact.
1741                **
1742                **       2) x/y is exact. If x/y is exact then it is represented
1743                **          by 15 digits, therefore IP(x/y) will just be some of
1744                **          its leading digits (i.e. exact).
1745                **
1746                **
1747                **      Signals inexact (SB=1) when :
1748                **
1749                **        |x/y| >= 1E14 and x/y is inexact. Unfortunately, x/y
1750                **        can satisfy these conditions and still have IP(x/y)
1751                **        be exact in rare cases.
1752                **
1753                **        Example:
1754                **
1755                **         1E25 div 9999999999  =  IP(10000000000100000.0000100..)
1756                **
1757                **          = 1.0000000001E15 exactly, even though |x/y|>1E14
1758                **
1759                **         and x/y is inexact.
1760                **
1761                **
1762                ****************************************************************
1763                ****************************************************************
1764 0C8F9 7730 =IDIV12 GOSUB  SPLTAC
1765 0C8FD 7BAB =IDIV15 GOSUB  DV2-15          Y/X
1766 0C901 831          ?XM=0                 no xcpt.
1767 0C904 40           GOYES  IDV30
1768 0C906 01           RTN                   XM=1, so preserve P=xcpt &
1769                                           C(A)=msg code (no need to
1770                                           CLRFRC).
1771
1772 0C908 78ED  IDV30  GOSUB  CLRFRC         IP(Y/X)
```

```
1773 0C90C 89F          ?P=    15             NaN,Inf or > 1E14 ?
1774 0C90F 00           RTNYES                Rtn with SB from  y/x.
1775
1776 0C911 822          SB=0                  Exact since y/x has =< 14
1777                                          digits in its int. part
1778 0C914 01           RTN
1779            * ------------------------------- *
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791            ************************************************************
1792            ************************************************************
1793            **
1794            ** Name:    SIGNAN  -  Signaling NaN
1795            **
1796            ** Category:   MATH
1797            **
1798            ** Purpose:
1799            **      To create a Signaling NaN (From exec'g the NAN fcn)
1800            **
1801            ** Entry:
1802            **      No entry conditions.
1803            **
1804            ** Exit:
1805            **      C=Signaling NaN (12 form) for user.
1806            **       =|0|0000|0...0|F02|
1807            **
1808            ** Calls:      none
1809            **
1810            ** Uses.......
1811            **  Inclusive: P; C
1812            **
1813            ** Stk lvls:   0
1814            **
1815            ************************************************************
1816            ************************************************************
1817 0C916      =SIGNAN
1818 0C916 AF2          C=0    W              clr msg code for Signal'g NaN
1819 0C919 20           P=     0
1820 0C91B 3220         LCHEX  #F02           C = |0|0000|000|00000|F02|
           F
1821 0C920 01           RTN
1822
1823
1824
1825
1826            ****************************
```

```
1827 0C922 8E00 =uAD12  GOSUBL =uMODES        copy rnd'g modes to st. bits
           00
1828 0C928 7800         GOSUB  SPLTAC
1829 0C92C 763A         GOSUB  AD15M
1830 0C930 6360         GOTO   uRES12
1831               ****************************
1832
1833
1834
1835
1836               ****************************************************************
1837               ****************************************************************
1838               ** Name:(S) SPLTAC  -  Split & normalize A & C
1839               **
1840               ** Category:   MTHUTL
1841               **
1842               ** Purpose:    Split & Normalize values in A & C.
1843               **
1844               ** Entry:      A:X  C:Y        [12-digit forms]
1845               **
1846               ** Exit:       A,B:X    C,D:Y  [15-digit forms]
1847               **
1848               ** Calls:      SPLITA,SPLITC
1849               **
1850               ** Alters (INC): A,B,C,D,Carry
1851               **
1852               ** Stk lvls:   0
1853               **
1854               ** History:
1855               **
1856               **    Date     Programmer              Modification
1857               **  --------   ----------   ----------------------------------
1858               **   6/28/82   SB           A field instead of W
1859               **   9/23/82   SB           This routine moved (eliminate GOTO)
1860               ****************************************************************
1861               ****************************************************************
1862 0C934 D7   =SPLTAC D=C    A
1863 0C936 07         C=RSTK                  Save Return
1864 0C938 738D       GOSUB  SPLITA
1865 0C93C 06         RSTK=C                  Restore Return
1866 0C93E DB         C=D    A
1867
1868               ****************************************************************
1869               ****************************************************************
1870               **
1871               ** Name:(S) SPLITC  -  SPLIT C
1872               **
1873               ** Category:   MTHUTL
1874               **
1875               ** Purpose:
1876               **      see SPLITA
1877               **
1878               ** Entry:
1879               **      C=x'  (external form)
1880               **
```

```
1881            ** Exit:
1882            **      (C,D)=x (normal internal form)
1883            **
1884            ** Calls:      (none)
1885            **
1886            ** Uses.......
1887            **  Inclusive: C,D
1888            **
1889            ** Stk lvls:   0
1890            **
1891            ** Detail:
1892            **      see SPLITA
1893            **
1894            ********************************************************************
1895            ********************************************************************
1896 0C940 ADF  =SPLITC CDEX   M
1897 0C943 AB7         D=C    X
1898 0C946 A26         C=C+C  XS
1899 0C949 D2          C=0    A
1900 0C94B 540         GONC   SPLC10
1901 0C94E CE          C=C-1  A
1902 0C950 ABB  SPLC10 C=D    X
1903 0C953 AB3         D=0    X
1904
1905 0C956 04   =NRMLCD SETHEX
1906 0C958 B26         C=C+1  XS
1907 0C95B A2E         C=C-1  XS
1908 0C95E 05          SETDEC
1909 0C960 5A0         GONC   SHFD
1910
1911 0C963 AD2         C=0    M              00Fxx for expon
1912 0C966 AA7         D=C    XS             make mant#0 for NaN & inf
1913 0C969 02          RTNSC
1914
1915 0C96B AC7  SHFD   D=C    S              Save C[S]=sign in D[S]
1916 0C96E 80FF        CPEX   15             Save P in C[S]
1917 0C972 2E          P=     14
1918 0C974 91B         ?D=0   WP
1919 0C977 01          GOYES  SHFD30
1920 0C979 90F  SHFD20 ?D#0   P
1921 0C97C D0          GOYES  SHFD40
1922 0C97E CE          C=C-1  A
1923 0C980 B93         DSL    WP
1924 0C983 65FF        GOTO   SHFD20
1925
1926 0C987 D2   SHFD30 C=0    A
1927 0C989 20   SHFD40 P=     0              Restore P
1928 0C98B 80FF        CPEX   15             Restore C[S]
1929 0C98F ACF         DCEX   S              D(S):=0
1930 0C992 03          RTNCC
1931
1932
1933
1934
1935
```

```
1936
1937
1938
1939          ****************************************************************
1940          ****************************************************************
1941          **
1942          ** Name:(S) uRES12  -  User Result
1943          ** Name:(S) uRESNX  -  User Result (non exceptional)
1944          ** Name:(S) uRESXT  -  User Result for exact results
1945          **
1946          ** Category:   MTHUTL
1947          **
1948          ** Purpose:
1949          **      To pack the 15-form input into a 12-form  result for
1950          **      delivery to the user. This includes rounding according
1951          **      to the user's mode, checking for xcpt'ns & consulting
1952          **      relevent trap values, setting the xcptn flags,and sending
1953          **      off any warning messages or errors. The external default
1954          **      result (12 form) is returned in reg C.
1955          **
1956          ** Entry:
1957          **    1.(A,B)&SB contain x (the unpacked result)
1958          **    2.XM is set if x is the result of an xcpt (DVZ or IVL)
1959          **       if XM=1 then P=(DZP,IVP or TYP0^0) tells which xcptn and
1960          **       C(A)=msg code (for specific xcptn e.g. 0/0,LOG(0),etc.)
1961          **    **Note**: DZP=3; IVP=4; TYP0^0=14.
1962          **
1963          **    3.D1=top math stk -- only used for a wrn. msg., to check
1964          **        avail.mem. for a possible mem err.
1965          **
1966          **    CODE:   =uRESNX  GOSUB  uRND>P
1967          **    ----    =uRESXT  GOSUB  HTRAP
1968          **                     GOSUB  HNDLFL
1969          **                     GOTO   MESSG
1970          **
1971          **
1972          ** Exit:
1973          **      C:=x' (the 12digit packed result).
1974          **      The XCPTN flags are set and any messages have been dis-
1975          **      played (including errors).
1976          **
1977          ** Calls:      uRND12, HTRAP, HNDLFL, MESSG
1978          **
1979          ** Uses.......
1980          **  Inclusive: P; A,B,C,D; R regs [3]; ST.[7..11];
1981          **             HD.ST.[SB,XM];
1982          **
1983          ** Stk lvls:   2  (provided that MFWRNQ uses <= 4 levs.)
1984          **
1985          ** NOTE:
1986          **      TYP0^0 "xcptns" (0^0 & Inf^0) return 1. They are not
1987          **      IVL xcptns but do consult the IVL trap. No flags are
1988          **      raised, but TRAP(IVL)#0 gives a wrn'g while =0 gives
1989          **      an error. XM=1 & P=14 signals TYP0^0 "xcptn".
1990          **
```

```
1991            ********************************************************************
1992            ********************************************************************
1993 0C994      =uRES12                         INPUT: X = (A,B)&SB
1994                                                   XM=1 when an xcptn has
1995                                                      occured (DZ,IV or TYP0^0)
1996                                                   P=<xcpt>  DZ,IVor14(if XM=1)
1997
1998                                            OUTPUT: C=RESULT, user's flags set
1999
2000
2001 0C994 831          ?XM=0                   NORMAL RESULT? (NO XCPT)
2002 0C997 42           GOYES  NRM12
2003
2004 0C999 7140         GOSUB  RNDXCP           C:=RESULT, sIX:=0 (IV,DZ&TYP0^0)
2005
2006 0C99D 893          ?P=    DZP              DVZ ?
2007 0C9A0 12           GOYES  uRESXT
2008 0C9A2 894          ?P=    IVP
2009 0C9A5 C1           GOYES  uRESXT
2010
2011            *   P=TYP0^0 (P=14)  *
2012 0C9A7 A83          D=0    P                Use imaginary 0^0 trap value of 0
2013                                             to get traps from HTRAP.
2014 0C9AA 7191         GOSUB  TRP30            on exit D(A)=TRAPS & B(S)=0
2015 0C9AE 24           P=     IVP
2016 0C9B0 908          ?D=0   P                Is TRAP(IVL)=0 ?
2017 0C9B3 61           GOYES  messg            disp error (0^0,inf^0)  *Here,
2018 0C9B5 A4D          B=B-1  S                disp warn (0^0,inf^0)   *NO flags
2019 0C9B8 401          GOC    messg            "goto"                    Set
2020
2021
2022 0C9BB 22    NRM12  P=     2                Round to 12 sig. digits
2023 0C9BD 7E00 =uRESNX GOSUB  uRND>P           Non Xcpt'l res. (needs rnd'g)
2024 0C9C1 7A61 =uRESXT GOSUB  HTRAP            Consult TRAPS
2025 0C9C5 7002         GOSUB  HNDLFL           HANDLE FLAG SETTING
2026 0C9C9 6D42 messg   GOTO   MESSG            DISP ANY WRN'G OR ERROR MESSAGE
2027
2028
2029
2030
2031
2032
2033            ********************************************************************
2034            ********************************************************************
2035            **
2036            ** Name:(S) uRND>P  -  user ROUND
2037            ** Name:(S) RND12+  -  Round 15-form
2038            ** Name:(S) OVFL    -  Create overflow value
2039            **
2040            ** Category:   MTHUTL
2041            **
2042            ** Purpose:
2043            **      To round an internal no. x to external form, according
2044            **       to the user's rounding mode.
2045            **      RND12+: Round according to status bits (s10,s11). Given
```

```
2046            **       by (0,0)=NEAR, (0,1)=ZERO, (1,0)=POS, (1,1)=NEG.
2047            **
2048            ** Entry:
2049            **      (A,B)&SB= x
2050            **      P=rounding position (e.g. P=2 for 12dig.;P=9 for 5dig.)
2051            **        2 <= P <= 13
2052            **      DEC MODE
2053            **
2054            ** Exit:
2055            **      C:= x' (rounded external form)
2056            **      sIX(s7):= inexact info.
2057            **      P:=OVP(2),UNP(1) or OKP(0)  (ovfl,unfl or ok)
2058            **      B(A)=msg code of OV or UN resp.
2059            **
2060            ** Calls:       NRMLAB, RNDNRM, BIASC+,-, BIASA-, HUGE20, BIG,
2061            **               uMODES
2062            **
2063            ** Uses.......
2064            **  Inclusive: P; A,B,C,D; R regs [3]; ST.[7(sIX),8..11];
2065            **             HD.ST.[SB];
2066            **
2067            ** Stk lvls:   1
2068            **
2069            ** NOTE:
2070            **      Original x is not always preserved !
2071            **      An inexact +/- 0 (i.e. SB=1) will be rounded to +/- 0
2072            **       with P=OKP and sIX(s7)=1 on exit.
2073            **
2074            ***************************************************************
2075            ***************************************************************
2076 0C9CD 22  =uRND12 P=     2              round to 12 digits
2077 0C9CF 8E00 =uRND>P GOSUBL =uMODES
               00
2078                                          SETS P= OV,UN or IX
2079
2080
2081                                          input: 1. rnd'g mode in st. bits
2082                                                 2. X=(A,B)
2083                                                 3. P=rnd'g position
2084                                          output: 1.     C=rnd12(X)
2085                                                  2. P in (OV,UN,IX), sIX
2086
2087
2088 0C9D5 7CFC =RND12+ GOSUB  NRMLAB
2089 0C9D9 521         GONC   RND50
2090
2091
2092            *  Round a NaN or Inf   *
2093 0C9DC 20          P=     OKP
2094 0C9DE 847  =RNDXCP ST=0   sIX
2095 0C9E1 AFD         BCEX   W              C:=MANT  (B:=MSG FOR uRES12)
2096 0C9E4 AB6         C=A    X              C(X)=F--
2097 0C9E7 AC6         C=A    S              C(S)=Sg
2098 0C9EA 01          RTN
2099
```

```
2100
2101
2102 0C9EC      RND50                          P set from uRND>P or RND>P entry.
2103 0C9EC 80CF        C=P    15               C(S):=P' (RND'G POSITION)
2104 0C9F0 AC5         B=C    S                B(S):=P' SAVED.
2105
2106 0C9F3 7AB0        GOSUB  RNDNRM           (RND100) ROUND NRM'L FINITE #
2107                                           (C,D)=rounded result, sIX correct
2108
2109           *  CHECK FOR EXCEPTIONS  OV,UN,IX
2110 0C9F7 103         R3=A                    SIGN & EXPON OF ORIGINAL X
2111 0C9FA 8E00        GOSUBL =BIASC+          BIAS EXPON BY 50000
           00
2112 0CA00 DA          A=C    A                A=BIASED RNDED EXPON
2113 0CA02 20          P=     0
2114 0CA04 3410        LCHEX  49501            MIN LEGAL EXPON (BIASED)
           594
2115 0CA0B 95D         ?B#0   M                MANT NON ZERO? (don't change
2116 0CA0E 80          GOYES  XCPT30             to W field -- B(S) has P' saved)
2117
2118           *   Pack (A,B)=0     *
2119                                           Note: P= OKP = 0 already set &
2120                                             sIX set approp'ly (RNDNRM).
2121
2122 0CA10 AF3         D=0    W                Zero mantissa in case RNDNRM
2123                                            made D=00...01 (ROUND to POS
2124                                            or NEG modes)
2125 0CA13 595         GONC   XCPT50           "GOTO" (clean the expon)
2126
2127
2128
2129 0CA16      XCPT30
2130 0CA16 8B2         ?A<C   A                EXP<MIN ?
2131 0CA19 81          GOYES  UNFL
2132
2133 0CA1B FA          C=-C   A                MAX LEGAL EXPON (biased !)
2134 0CA1D 8B6         ?A>C   A                EXPON>MAX ?
2135 0CA20 35          GOYES  OVFL
2136
2137 0CA22 8E00        GOSUBL =BIASA-          UNBIAS
           00
2138 0CA28 D6          C=A    A
2139
2140 0CA2A 20          P=     OKP              sIX already set
2141
2142 0CA2C ADB  XCPT55 C=D    M
2143 0CA2F 01          RTN
2144
2145
2146
2147           *  0 < |Result| < EPS  *
2148 0CA31      UNFL
2149 0CA31 8E00        GOSUBL =BIASC-          UNBIAS MIN EXPON   (99501)
           00
2150 0CA37 113         A=R3                    RESTORE ORIGINAL EXPON FOR # SHIFTS
```

```
2151 0CA3A DE          ACEX   A            A=MIN UNBIASED EXPON (OF DENOR. #)
2152 0CA3C EE          C=A-C  A            # SHIFTS
2153 0CA3E 2E          P=     14
2154
2155 0CA40 CE   UNFL10 C=C-1  A
2156 0CA42 4A0         GOC    UNFL20
2157 0CA45 B95         BSR    WP           (ALSO TAKES CARE OF SB FOR NEXT RND)
2158 0CA48 0D          P=P-1
2159 0CA4A 55F         GONC   UNFL10
2160                                       May fall through with inex 0 (0E-499
2161                                        & SB=1). This is ok for RNDNRM.
2162
2163 0CA4D AC9  UNFL20 C=B    S
2164 0CA50 80DF        P=C    15           RESTORE P=P' for rnd'g.
2165
2166 0CA54 7950        GOSUB  RNDNRM
2167
2168           *  Signal "tiny" result, UNF will be determined in HTRAP   *
2169 0CA58 21          P=     UNP          NOTE: sIX may be 0 or 1 here .
2170
2171 0CA5A 133         AD1EX
2172 0CA5D 1E00        D1=(4) =eUNFLW      UN MSG CODE
          00
2173 0CA63 133         AD1EX
2174
2175 0CA66 D8          B=A    A            B(A):=eUNFLW
2176 0CA68      UNFL40
2177 0CA68 97F         ?D#0   W            IS ANSWER #0?
2178 0CA6B 1C          GOYES  XCPT55
2179
2180 0CA6D D2   XCPT50 C=0    A            "C=0  X"  (EXPON OF 12DIG ANS)
2181 0CA6F 6CBF        GOTO   XCPT55
2182
2183
2184           *   OVFL assumes P=0 on entry     *
2185                                       Here the Sign is in A(S)&C(S).
2186 0CA73      =OVFL                       called in AB&ASN to set for uRESNX
2187 0CA73 857         ST=1   sIX
2188 0CA76 3300        LC(4)  =eOVFLW      OV MSG CODE
          00
2189
2190 0CA7C D5          B=C    A            B(A):=eOVFLW
2191 0CA7E 22          P=     OVP
2192 0CA80 87A         ?ST=1  sINFRD       RND TO +/- INF
2193 0CA83 31          GOYES  OVFL20
2194
2195 0CA85 87B         ?ST=1  sNEGRD       Round to 0 ?
2196 0CA88 02          GOYES  OVFL40
2197
2198
2199           *  RND TO NEAREST   *
2200           *  Create Inf Result  *
2201 0CA8A      OVINF                       DON'T CHANGE P
2202 0CA8A AF2         C=0    W
2203 0CA8D AC6         C=A    S
```

```
2204 0CA90 8C00          GOLONG =HUGE20          +/- INF.
           00
2205
2206
2207               *   RND TO +/- INF  *
2208 0CA96         OVFL20
2209               *                          <<<  sIX already set
2210 0CA96 87B             ?ST=1  sNEGRD          TO -INF ?
2211 0CA99 A0              GOYES  OVFL30
2212
2213                                              TO +INF
2214               *   To +Inf   *
2215 0CA9B 94E             ?C#0   S
2216 0CA9E A0              GOYES  OVFL40          TO +INF & NEG OVFL
2217
2218 0CAA0 59E             GONC   OVINF           TO +INF & POS OVFL
2219
2220
2221               *   To -Inf   *
2222 0CAA3 94E  OVFL30     ?C#0   S               To -inf & neg OVFL ?
2223 0CAA6 4E              GOYES  OVINF
2224                                              To -inf & pos OVFL
2225               *   Create MAXREAL Result  *
2226 0CAA8         OVFL40                         LEAVE SIGN ALONE
2227 0CAA8 AD2             C=0    M               (FOR =BIG WITHOUT SIGN)
2228 0CAAB 8C00  big       GOLONG =BIG            +/- 9.99...99E499  (WON'T ALTER P)
           00
2229
2230               *----------------------*
2231
2232
2233
2234               ************************************************************************
2235               ************************************************************************
2236               **
2237               ** Name:(S) RNDNRM  -  Round a Normal Number
2238               **
2239               ** Category:   MTHUTL
2240               **
2241               ** Purpose:
2242               **      To round the mantissa of a finite internal no. x,
2243               **      according to the rounding modes specified.
2244               **
2245               ** Entry:
2246               **      (A,B)&SB = x
2247               **      P=rounding position (e.g. P=2 for 12 digit round;
2248               **        P=9 for 5 digit round)  0<=P<=14
2249               **      sINFRD(s10)&sNEGRD(s11) set for rounding mode
2250               **        (see =uMODES)
2251               **      DECMODE
2252               **
2253               ** Exit:
2254               **      (C,D) = x' (rounded value)   (and D[S]=0)
2255               **      sIX(s7) set iff the rounded result is inexact
2256               **      P=0
```

```
2257            **
2258            ** Calls:      None
2259            **
2260            ** Uses.......
2261            **  Inclusive: P; C,D; ST.[sIX(s7)];
2262            **
2263            ** Stk lvls:   0
2264            **
2265            ** NOTE:
2266            **    With an input of inex 0 in Rnd to Inf mode, the mantissa
2267            **    is rounded to 00...01 and its exponent is unchanged. In
2268            **    the other rounding modes the mantissa remains 0.
2269            **
2270            ********************************************************************
2271            ********************************************************************
2272
2273 0CAB1      =RNDNRM                          RND FINITE #  at position P
2274 0CAB1 AF9          C=B    W
2275 0CAB4 AC2          C=0    S                 MAKES D=0 W  FOR ZERO RESULT.
2276 0CAB7 847          ST=0   sIX               INIT. IX  BIT
2277 0CABA 91E          ?C#0   WP                DEFINITELY INEXACT?
2278 0CABD 70           GOYES  RND120
2279 0CABF 832          ?SB=0                    EXACT?
2280 0CAC2 C2           GOYES  RND150
2281
2282            *  Inexact Result   *
2283 0CAC4 857  RND120  ST=1   sIX               SET IX  RND FLAG
2284 0CAC7 87A          ?ST=1  sINFRD            RND TOWARD INF
2285 0CACA C4           GOYES  RND210
2286 0CACC 87B          ?ST=1  sNEGRD            TO 0
2287 0CACF 80           GOYES  RND125
2288
2289
2290            *   Round to Nearest   *
2291 0CAD1 A16          C=C+C  WP                RND TO NEAREST
2292 0CAD4 490          GOC    RND130
2293
2294
2295            ***  TRUNCATE   ***
2296
2297 0CAD7 A92  RND125  C=0    WP                Truncate.
2298 0CADA 6310         GOTO   RND150
2299
2300
2301 0CADE 91E  RND130  ?C#0   WP                DEFINITELY RND UP?
2302 0CAE1 70           GOYES  RND140
2303 0CAE3 832          ?SB=0                    HALF WAY. STICKY BIT 0 ?
2304 0CAE6 22           GOYES  RND190
2305
2306
2307
2308
2309            ***  ROUND UP  ***
2310
2311 0CAE8 A92  RND140  C=0    WP                RND UP.
```

```
2312 0CAEB B9E          C=-C-1 WP            GUARD DIGITS ALL 9'S
2313
2314
2315            *** Perform the Rounding *** At this point, guard digits will
2316                                         be all 9s (Up) or all 0s (Trunc).
2317 0CAEE 2E   RND150  P=     14
2318 0CAF0 AF7          D=C    W
2319 0CAF3 AF6          C=A    W
2320 0CAF6 B17          D=D+1  WP            RND UP. CARRY?
2321 0CAF9 570          GONC   RND160        NO.
2322 0CAFC E6           C=C+1  A             YES. INC EXPON
2323 0CAFE B07          D=D+1  P             MAN=1000...
2324 0CB01 20   RND160  P=     0
2325 0CB03 A83          D=0    P             CLEAR POSSIBLE #0 LSD
2326 0CB06 01           RTN
2327
2328
2329
2330 0CB08      RND190                       NOTE: C=0 WP here.
2331 0CB08 AF7          D=C    W
2332 0CB0B 81F          DSRB                 Does not alter SB.
2333 0CB0E 90B          ?D=0   P             SET CARRY IF EVEN
2334 0CB11 DD           GOYES  RND150        EVEN ALREADY NO RND
2335 0CB13 54D          GONC   RND140        ODD. RND UP TO EVEN
2336
2337            *   Infinity Rounds   *
2338 0CB16 87B  RND210  ?ST=1  sNEGRD        RND TO -INF
2339 0CB19 A0           GOYES  RND220
2340
2341            *   To +INF    *
2342 0CB1B 948          ?A=0   S
2343 0CB1E AC           GOYES  RND140        RND UP
2344 0CB20 56B          GONC   RND125        NEG. TRUNCATE
2345
2346            *   To -INF    *
2347 0CB23 948  RND220  ?A=0   S
2348 0CB26 1B           GOYES  RND125        TRUNC
2349 0CB28 5FB          GONC   RND140        NEG. RND UP
2350
2351
2352 0CB2B 639B splitA GOTO    SPLITA
2353
2354
2355
2356            **************TRAPPING MECHANISM*********************
2357
2358
2359
2360
2361
2362
2363            ************************************************************
2364            ************************************************************
2365            **
2366            ** Name:(S) HTRAP   -  HANDLE TRAPS
```

```
2367            **
2368            ** Category:   MTHUTL
2369            **
2370            ** Purpose:
2371            **      To determine any trapping action that is specified (e.g.
2372            **      alter the IEEE default result or halt) on an xcptn .
2373            **
2374            ** Entry:
2375            **      C= x' (the 12 form IEEE default result)
2376            **      Trap to be checked is indicated by:
2377            **        P=<xcpt> {OK,UN,OV,DZ or IV}
2378            **        sIX(s7)= xact/inex info  (esp. for P=OK)
2379            **      B(A)=msg code (for UN,OV,DZ or IV only)
2380            **
2381            ** Exit:
2382            **      C:= x'' (revised result --after consulting traps)
2383            **      B(S):= 0 for an error (HALT) ;  9 otherwise (continue)
2384            **      B(A)=msg code for IX,UN,OV,DZ or IV.
2385            **      P = updated xcpt.
2386            **      sIX(s7) reflects updated exact/inexact info
2387            **      Sets DECMODE (only when GOSUB BIG is executed)
2388            **
2389            **   Special exit condition:
2390            **      (preserved for USGOVF -- DISP USING OVFL)
2391            **
2392            **      Whenever HTRAP exits with B(S)=0 (i.e. Halt)
2393            **      then:
2394            **
2395            **      1) If TRAP(OVF) caused the Halt then s7 (sIX) is
2396            **         NOT altered from its entry state.
2397            **
2398            **      2) If TRAP(INX) caused the Halt then s7 (sIX) is
2399            **         set to 1 on exit.
2400            **
2401            **
2402            **
2403            ** Uses.......
2404            **  Inclusive: A,B,C,D; ST.[sIX(s7)];
2405            **
2406            ** Calls: BIG
2407            **
2408            ** Stk lvls:   1
2409            **
2410            ** NOTE:
2411            **
2412            **
2413            ****************************************************************
2414            ****************************************************************
2415 0CB2F      =HTRAP
2416 0CB2F AC1          B=0    S
2417 0CB32 A4D          B=B-1  S              init.B#0 (i.e. WRN not ERR message)
2418 0CB35 880          ?P#    OKP            NOT(IX or XACT) ?
2419 0CB38 70           GOYES  TRP30
2420
2421 0CB3A 867          ?ST=0  sIX            XACT?
```

```
2422 0CB3D 00          RTNYES
2423
2424
2425 0CB3F      TRP30
2426 0CB3F AFF         CDEX   W              D:=x' (C may be D-input from uRES12
2427                                               on entry to TRP30. P=14 case)
2428 0CB42 133         AD1EX                 save D1 into A
2429
2430           *  get TRAPS  *
2431 0CB45 1F00        D1=(5) =TRPREG        addr(IX) - 1
           000
2432 0CB4C 147         C=DAT1 A              C(A):=TRAPS[IV,DZ,OV,UN,OK]
2433 0CB4F 133         AD1EX
2434
2435 0CB52 80CF        C=P    15             C(S):=xcpt=P now use P
2436                                             as the CURRENT xception
2437
2438           *  Test TRAP(P)  *
2439 0CB56      TSTRAP
2440
2441 0CB56 A0E         C=C-1  P
2442 0CB59 4E0         GOC    TRP=0
2443 0CB5C A0E         C=C-1  P
2444 0CB5F 404         GOC    TRP=1
2445 0CB62 A0E         C=C-1  P
2446 0CB65 461         GOC    TRP=2
2447
2448           *       not a mainframe trap
2449
2450 0CB68      TRP=0                        ***HALT***
2451 0CB68 880         ?P#    OKP            NOT IX TRAP ?
2452 0CB6B A0          GOYES  ERRMSG
2453
2454           *    IX halt  (P=0)    *
2455 0CB6D D2          C=0    A
2456 0CB6F 3100        LC(2)  =eINX
2457 0CB73 D5          B=C    A              B(A):= eINX
2458 0CB75      ERRMSG
2459 0CB75 AC1         B=0    S
2460 0CB78 6800        GOTO   RTNDEF
2461
2462
2463
2464 0CB7C      TRP=2                        ***IEEE***
2465 0CB7C 880         ?P#    OKP            current TRAP checked is not IX ?
2466 0CB7F B0          GOYES  CKIX?
2467
2468 0CB81 80DF RTNDEF P=C    15             restore xcpt  (P:=xcpt)
2469 0CB85 AFF         CDEX   W              C=x & sometimes D=TRAPvalues.
2470 0CB88 01          RTN
2471
2472 0CB8A      CKIX?
2473 0CB8A 877         ?ST=1  sIX            Inexact ?
2474 0CB8D D0          GOYES  CKIXP
2475 0CB8F 881         ?P#    UNP
```

```
2476 0CB92 FE           GOYES  RTNDEF          Exact & not UNF
2477
2478            *   "Tiny" & Exact  (i.e. rslt<EPS & exact)    *
2479 0CB94 AC2          C=0    S               xcpt:=0KP (no UNF--new defn of UNF)
2480 0CB97 59E          GONC   RTNDEF          "goto"
2481
2482
2483 0CB9A 20   CKIXP   P=     0KP             to check TRAP(IX)
2484 0CB9C 69BF         GOTO   TSTRAP
2485
2486
2487
2488
2489 0CBA0      TRP=1                          ***TRADITIONAL FINITE DEFAULTS (75C)
2490 0CBA0 890          ?P=    0KP              was TRAP(IX) checked?
2491 0CBA3 ED           GOYES  RTNDEF
2492
2493 0CBA5 894          ?P=    IVP
2494 0CBA8 DC           GOYES  ERRMSG
2495
2496 0CBAA 857          ST=1   sIX             (UN,0V and DZ defaults are inex)
2497 0CBAD AFA          A=C    W               save trapvalues & original xcpt
2498 0CBB0 AF2          C=0    W
2499 0CBB3 891          ?P=    UNP             Flush to 0?
2500 0CBB6 60           GOYES  GETSGN
2501
2502            *    P=0V or DZ  --- create MAXREAL   *
2503
2504 0CBB8 7FEE         GOSUB  big             9...9E499 (DOES NOT USE P,Sets DEC)
2505
2506 0CBBC ACB  GETSGN  C=D    S               +/- for UN,0V or DZ
2507 0CBBF AF7          D=C    W               D:=DEF. VAL.
2508 0CBC2 AF6          C=A    W               restore trapvalues & xcpt
2509 0CBC5 64DF         GOTO   CKIXP
2510
2511
2512
2513
2514
2515
2516
2517
2518            ****************************************************************
2519            ****************************************************************
2520            **
2521            ** Name:(S) HNDLFL  -  HANDLE FLAG SETTING
2522            **
2523            ** Category:   MTHUTL
2524            **
2525            ** Purpose:
2526            **      To set user's xcptn flags (all at once).
2527            **
2528            ** Entry:
2529            **      P=<xcpt>, <xcpt> in {0K,UN,0V,DZ,IV}.
2530            **      sIX(s7)= inex info.
```

```
2531                **
2532                ** Exit:
2533                **      user's xcptn flags have been updated.
2534                **      D(X) will contain bit mask of xcptns set (b11 to b7
2535                **        represents IV,DZ,OV,UN,IX)
2536                **
2537                ** Uses.......
2538                **  Inclusive: A(A),D(X); R regs [3];
2539                **
2540                ** Stk lvls:   0
2541                **
2542                ** NOTE:
2543                **      The info. from HTRAP [C,B(S),B(A),P & sIX] is preserved.
2544                **
2545                ********************************************************************
2546                ********************************************************************
2547 0CBC9          =HNDLFL
2548 0CBC9 10B            R3=C                    SAVE C
2549 0CBCC AB2            C=0     X
2550 0CBCF 877            ?ST=1   sIX
2551 0CBD2 A0             GOYES   HNDL20          (inex. & possibly another xcptn)
2552 0CBD4 890            ?P=     0KP
2553 0CBD7 B3             GOYES   RSTDEF          EXACT RESULT (RTN)
2554 0CBD9 580            GONC    HNDL30          "GOTO". (other xcptn & exact)
2555
2556 0CBDC B26  =HNDL20   C=C+1   XS
2557 0CBDF 81E            CSRB                    SET IX BIT IN C[b7].
2558
2559
2560
2561                *                                                       *
2562                *   Set C[XS] so that C[b11]...C[b8] -->  IVL...UNF     *
2563                *                                                       *
2564                *      xcpt   P   C[XS]                                 *
2565                *      ----   -   -----                                 *
2566                *      IVL    4     8                                   *
2567                *      DVZ    3     4                                   *
2568                *      OVF    2     2                                   *
2569                *      UNF    1     1                                   *
2570                *      INX    0     0  with C[b7]=1.                    *
2571                *                                                       *
2572                *                                                       *
2573
2574
2575
2576
2577 0CBE2 80C2 HNDL30   C=P     2               C(XS):=P =XCPTN
2578 0CBE6 894            ?P=     IVP
2579 0CBE9 A0             GOYES   HNDL40
2580 0CBEB 883            ?P#     DZP             i.e. OV,UN or illegal P.
2581 0CBEE 80             GOYES   XFSET
2582 0CBF0 A2E            C=C-1   XS              (DZ -> 2)
2583 0CBF3 A26  HNDL40    C=C+C   XS              (shift left 1 bit for DZ & IV)
2584
2585
```

```
2586            *  Set the RAM exception flags   *
2587
2588 0CBF6 133  XFSET   AD1EX
2589 0CBF9 1F00         D1=(5) (=SYSFLG)-1
          000
2590 0CC00 AB7          D=C    X              D(X):= XCPTN BITS FOR MASK
2591 0CC03 1573         C=DAT1 X              xcptn flags (IV -> IX start'g in XS
2592 0CC07 0E3F         C=C!D  X              OR to set sticky flags
2593 0CC0B 1553         DAT1=C X              COPY TO USER'S FLAGS.
2594 0CC0F 131          D1=A                  RESTORE SYS. D1
2595 0CC12 11B  RSTDEF  C=R3                  Restore C
2596 0CC15 01           RTN
2597
2598
2599
2600
2601
2602
2603
2604
2605            ********************************************************************
2606            ********************************************************************
2607            **
2608            ** Name:(S) MESSG   -  MESSAGE
2609            **
2610            ** Category:   MTHUTL
2611            **
2612            ** Purpose:
2613            **      To display a warning message without disturbing most
2614            **      of the CPU or Math Scratch Stack. It uses available
2615            **      memory instead, to preserve C,R0,R1,R2,R4, D0,D1, Status
2616            **      Bits, Math Scratch (=SCRST0) and RSTK levels.
2617            **
2618            ** Entry:
2619            **   1) B(A)=msg code; B(W) used if msg has text insertion (see
2620            **        MFWRNQ).
2621            **   2) B(S)= 0 for error
2622            **          = 9 otherwise
2623            **   3) If B(S)=9 then
2624            **        P=0 ==> no msg   (used to supress msg )
2625            **        P#0 ==> put out warning msg
2626            **
2627            **   4) D1=top of math stk (end of available memory)
2628            **           -- used only for mem chk when a warning is sent out.
2629            **
2630            **
2631            ** Exit:
2632            **      Displays warn/err msg & rtns to main driver on an err.
2633            **
2634            ** Calls:      MFWRNQ or exits thru BSERR, CHKmem, SNAPLC,
2635            **             MOVEU3, MOVED3, SNAPR* .
2636            **
2637            ** Uses.......
2638            **  Inclusive: P; A,B,D;R3; (unless an error occurs--BSERR)
2639            **     The Math Scratch Area is saved to Available Memory
```

```
2640            **      since the display routines check Service Request and
2641            **      an Alarm calculation uses math scr.
2642            **
2643            ** Stk lvls:  2      1+[Levels(MFWRNQ) - 2(saved Levels)]
2644            **
2645            ****************************************************************
2646            ****************************************************************
2647 0CC17      =MESSG                          Messages (esp. for Math Routines)
2648 0CC17 94D          ?B#0   S                Don't HALT ?
2649 0CC1A C0           GOYES  NOHALT
2650
2651            *  HALT on ERROR  *
2652 0CC1C D9           C=B    A                C(A)=ERR MSG
2653 0CC1E 20           P=     0                std. err msg parameter
2654 0CC20 8C00 mfERR   GOLONG =BSERR           BASIC ERROR (MFERR)
       00
2655
2656
2657 0CC26 890   NOHALT ?P=    OKP              IX or XACT ( WITH NO HALT)
2658 0CC29 00           RTNYES                  rtn promptly (no msg)
2659
2660            *   attempt warning msg (observe QUIET)       *
2661            *   To save code, QUIET will not be checked   *
2662            *   before saving away MathSCR and CPU regs.  *
2663            *   (QUIET check would involve saving ptrs    *
2664            *    and doing a POLL on the WARN'G)          *
2665
2666
2667
2668            *  Save CPU info. and Math Scratch Stack (MSCR) mantissas
2669            *  to available memory.
2670            *
2671            *   NOTATION for comments:
2672            *
2673            *
2674            *
2675            *  low --> high mem
2676            *
2677            *      MEMS       CPUST  SUBST  CPUEND    d1         MEME
2678            *   ---------------------------------------------------------
2679            * ...   |          |  CPU .(RSTK)| MSCR  |////////////|//////...
2680            *   ---------------------------------------------------------
2681            *
2682            *
2683            *
2684
2685
2686
2687 0CC2B 04           SETHEX                  hex arith. below on addrs.
2688 0CC2D 20           P=     0                for LC instrs.
2689 0CC2F 10B          R3=C                    R3=C (preserve C)
2690
2691            *  set up for CHKmem   *
2692 0CC32 137          CD1EX                   C=d1
2693 0CC35 D7           D=C    A
```

```
2694 0CC37 DA          A=C    A            A=D=d1
2695 0CC39 D2          C=0    A
2696 0CC3B 3147        LC(2)  52+64        length of CPU Area + Math Scr. Area
2697
2698 0CC3F 8F00        GOSBVL =CHKmem      not enough memory?
          000
2699 0CC46 49D         GOC    mfERR        Yes
2700
2701 0CC49       NOERR
2702
2703           *  set up for SNAPLC save to avail memory  *
2704                                       A=MEMS (from CHKmem), C=CPUST-MEMS
2705 0CC49 C2          C=A+C  A            C=CPUST
2706 0CC4B 135         D1=C
2707 0CC4E 119         C=R1                C=r1
2708 0CC51 AFF         CDEX   W            D=r1, C=d1
2709 0CC54 110         A=R0                A=r0
2710
2711 0CC57 8F00        GOSBVL =SNAPLC      save r0, r1,  d1 , d0 & msg - 47nbs
          000
2712                                        i.e. A, D , C(A), D0 & B(A)
2713
2714
2715           *  save 2 RSTK levels --  10nibs  *
2716 0CC5E 135         D1=C                D1 -> msg  (write over msg)
2717                                        i.e. D1=SUBST (start of rstk area)
2718 0CC61 2E          P=     14
2719 0CC63 07    SVSUBR  C=RSTK
2720 0CC65 145         DAT1=C A
2721 0CC68 174         D1=D1+ 5
2722 0CC6B 0C          P=P+1
2723 0CC6D 55F         GONC   SVSUBR
2724                                       D1 -> CPUEND (start of destination
2725                                           for MOVEU3)
2726
2727
2728           *
2729           *
2730           *         CPUST        SUBST     CPUEND         d1
2731           *     ---------------------------------------------------
2732           *          |r0,r1,d1,d0.rstk0,rstk1|???????????|/////////
2733           *     ---------------------------------------------------
2734           *                                    ^
2735           *                                    D1
2736           *
2737
2738
2739
2740
2741           *  set up for MOVEU3 -- to store MSCR area   *
2742 0CC70 1B00        D0=(5) =SCRST0      start of source (MSCR)
          000
2743 0CC77 D2          C=0    A            (P=0 from SVSUBR)
2744 0CC79 3104        LC(2)  64           MOVE 64 NIBS OF MSCR
2745 0CC7D 8E00        GOSUBL =MOveu3      MOVE MEM.
```

```
          00
2746
2747            *  update AVMEME to CPUST  and  save old MEME into R1  *
2748 0CC83 133       AD1EX                 A=d1
2749 0CC86 D2        C=0    A
2750 0CC88 3147      LC(2)  52+64          Length of save area
2751 0CC8C EA        A=A-C  A              A=CPUST=d1-(52+64)
2752 0CC8E 7560      GOSUB  GMEME          sets D0 -> AVMEME
2753 0CC92 109       R1=C                  R1=MEME
2754 0CC95 140       DAT0=A A              AVMEME:=CPUST
2755
2756
2757
2758
2759            *
2760            *       CPUST   CPUEND    d1          MEME
2761            *     ------------------------------------------------
2762            *        | CPU   | MSCR  |////////////|////////////
2763            *     ------------------------------------------------
2764            *        ^
2765            *       AVMEME
2766            *
2767
2768
2769
2770
2771
2772            *  Put out warning message  *
2773 0CC98 AF9       C=B    W              restore msg code
2774 0CC9B 28        P=     8
2775 0CC9D 8E00      GOSUBL =MFWRNQ        cks for QUIET not ON ERROR though
          00
2776                                         also updates ERRN & ERRL.
2777
2778            *  restore AVMEME -> MEME   *
2779 0CCA3 7050      GOSUB  GMEME          C=CPUST, D0 -> AVMEME
2780 0CCA7 111       A=R1                  A=MEME
2781 0CCAA 140       DAT0=A A              AVMEME -> MEME
2782
2783            *  Set up for MOVED3 -- restore MSCR  *
2784 0CCAD D5        B=C    A              B=CPUST
2785 0CCAF D2        C=0    A
2786 0CCB1 3147      LC(2)  52+64          (P=0 from MFWRNQ)
2787 0CCB5 C9        C=B+C  A              C=d1 =CPUST+(57+64)
2788 0CCB7 134       D0=C                  D0=d1 (end of source)
2789 0CCBA 1F00      D1=(5) (=SCRST0)+64   end of destination
          000
2790 0CCC1 D2        C=0    A
2791 0CCC3 3104      LC(2)  64             #NIBS OF MSCR TO BE MOVED (ONLY
2792                                        MANTISSAS)
2793 0CCC7 8E00      GOSUBL =M0ved3        moved MSCR back to SCRST0
          00
2794
2795            *  Restore the 2 saved RSTK levels  *
2796 0CCCD 2E        P=     14
```

```
2797 0CCCF 184  RSUBR   D0=D0- 5              D0 -> rstk(i)
2798 0CCD2 146          C=DAT0 A
2799 0CCD5 06           RSTK=C
2800 0CCD7 0C           P=P+1
2801 0CCD9 55F          GONC   RSUBR
2802
2803            *  Restore CPU info. from avail memory  *
2804                                          D0 -> rstk_0 Area
2805 0CCDC 136          CDOEX                C:=D0
2806 0CCDF 135          D1=C                 D1 -> MSG AREA for SNAPR*
2807 0CCE2 8E00         GOSUBL =snapr*
         00
2808 0CCE8 100          R0=A                 RESTORE R0
2809 0CCEB AFB          C=D    W             C=r1
2810 0CCEE 109          R1=C                 Restore R1=r1
2811
2812 0CCF1 05           SETDEC
2813 0CCF3 6E1F         GOTO   RSTDEF        restore C (C:=R3 & RTN)
2814
2815
2816
2817
2818 0CCF7       =GMEME                        GET AVMEME
2819 0CCF7 1B00         D0=(5) =AVMEME
         000
2820 0CCFE 146          C=DAT0 A
2821 0CD01 01           RTN
2822
2823
2824
2825
2826            *============================================================
2827
2828
2829
2830            ****************************************************************
2831            ****************************************************************
2832            **
2833            ** Name:(S) FINITA  -  Is (A,B) non-finite ?
2834            ** Name:(S) FINITC  -  Is (C,D) non-finite ?
2835            **
2836            ** Category:   MTHUTL
2837            **
2838            ** Purpose:
2839            **      To test for finite arguments.
2840            **
2841            ** Entry:
2842            **      FINITA: 15-form in AB
2843            **      FINITC: 15-form in CD
2844            **
2845            ** Exit:
2846            **      DEC Mode
2847            **      Carry Set indicates non-finite
2848            **      Carry Clear indicates finite
2849            **
```

```
2850                **
2851                ** Calls:      (None)
2852                **
2853                ** Uses.......
2854                **  Inclusive: Nothing
2855                **
2856                ** Stk lvls:   0
2857                **
2858                ****************************************************************
2859                ****************************************************************
2860
2861 0CD03         =FINITA
2862 0CD03 04              SETHEX
2863 0CD05 B24           A=A+1  XS
2864 0CD08 A2C           A=A-1  XS
2865 0CD0B 05            SETDEC
2866 0CD0D 01            RTN                    (carry is set for non-finite #s
2867                                               and clear for finite #s)
2868
2869
2870 0CD0F         =FINITC
2871 0CD0F 04            SETHEX
2872 0CD11 B26           C=C+1  XS
2873 0CD14 A2E           C=C-1  XS
2874 0CD17 05            SETDEC
2875 0CD19 01            RTN                    (carry set => non-finite
2876                                                clear => finite )
2877
2878                ************************************************
2879                ***                    LNSUB
2880                ***
2881                ***       INPUT: A= SIGN, EXP OF X
2882                ***              B= MANTISSA OF X
2883                ***         Restriction:  0 <= |X| < 1  ?
2884                ***       OUTPUT:Returns X/(|X|+1) in (A,B)
2885                ***       DESTROYS: A(X), B,C,D,P,SB
2886                ***       SUBROUTINE LEVELS: 0
2887                ************************************************
2888 0CD1B AF9   LNSUB  C=B     W
2889 0CD1E AF7           D=C     W
2890 0CD21 D8            B=A     A
2891 0CD23 BF7   LNSUB1 DSR     W
2892 0CD26 97B           ?D=0    W
2893 0CD29 21            GOYES   LNSUB2
2894 0CD2B E5            B=B+1   A
2895 0CD2D 55F           GONC    LNSUB1
2896 0CD30 2E            P=      14
2897 0CD32 B07           D=D+1   P
2898 0CD35 8CEC  LNSUB3 GOLONG DIV15           (OK, will not reset XM)
          7F
2899 0CD3B AF5   LNSUB2 B=C     W
2900 0CD3E 01            RTN
2901
2902
2903                ************************************************************
```

```
2904            ***                 LN1+X
2905            ***
2906            ***         INPUT:  A= SIGN,EXP OF X
2907            ***                 B= MANTISSA OF X
2908            ***         OUTPUT: RETURNS LN(1+X) IN STANDARD FORMAT
2909            ***         DESTROYS: A,B,C,D,P,SB,S10,R0
2910            ***         SUBROUTINE LEVELS: 1
2911            ************************************************************
2912
2913            ********************************************************************
2914            ********************************************************************
2915            **
2916            ** Name:(S) LN1+15  -  LN(1+X)
2917            ** Name:(S) LN1+XF  -  LN(1+X) for finite args only
2918            **
2919            ** Category:   MATH
2920            **
2921            ** Purpose:
2922            **      To compute  ln(1+x) from x.
2923            **
2924            ** Entry:
2925            **      Standard floating point math input.
2926            **
2927            ** Exit:
2928            **      Standard floating point math output.
2929            **      XM=1 & P=3  implies  LN(0)
2930            **           & P=4     "     LN(negative)
2931            **
2932            **
2933            ** Calls:      ADDONE, LN15,
2934            **
2935            ** Uses.......
2936            **  Inclusive: P; A,B,C,D; R regs [0]; ST.[s10];
2937            **                HD.ST.[SB,XM];
2938            **
2939            ** Stk lvls:   1
2940            **
2941            ********************************************************************
2942            ********************************************************************
2943 0CD40 77ED =LN1+12 GOSUB  splitA
2944
2945 0CD44 821  =LN1+15 XM=0
2946 0CD47 7576        GOSUB  FNPWDS        (Also does SB=0.)
2947 0CD4B 550         GONC   LN1+XF        finite arg.
2948 0CD4E 4C3         GOC    LNINF         "GOTO".Same result as LN(inf)
2949
2950 0CD51 84A  =LN1+XF ST=0   10           RESET NEGATIVE EXP FLAG
2951 0CD54 AC1         B=0    S
2952 0CD57 D6          C=A    A
2953 0CD59 E6          C=C+1  A
2954 0CD5B C6   =LN1+X4 C=C+C  A            EXPX<-1?
2955 0CD5D 4C0         GOC    LN1+X1        Yes ( |x| < 0.1 )
2956
2957
2958
```

```
2959
2960            *  [x+1] =< 0.9  or  [x+1] >= 1.1   *
2961
2962 0CD60 8EAC  LN1+X2 GOSUBL ADDONE          COMPUTE X+1,(rnd'g bits input don't
       5F
2963                                          matter since LN(+/-0)=-Inf )
2964                                          LN1+x already weeded xcptns.
2965 0CD66 6530         GOTO   LN30           LN(x+1)
2966
2967
2968
2969
2970
2971
2972
2973
2974            *   0.9 < [x+1] < 1.1   *
2975
2976 0CD6A 948   LN1+X1 ?A=0   S              [x+1] > 1 ?  (x>0?)
2977 0CD6D 80           GOYES  LN1+X5
2978
2979            *    x<0   *
2980 0CD6F 85A          ST=1   10             neg expon flag := true
2981 0CD72 560          GONC   ln1+X6         "GOTO".DECOMP prep on x (x<0)
2982
2983            *    x>=0  *
2984 0CD75 72AF  LN1+X5 GOSUB  LNSUB          FORM X/(X+1)
2985 0CD79 6A70  ln1+X6 GOTO   LN1+X6
2986
2987
2988
2989
2990
2991
2992
2993            ****************************************************************
2994            ****************************************************************
2995            **
2996            ** Name:(S) LN15    -  Natural Logarithm
2997            ** Name:(S) LN12    -  LOG for 12-form args.
2998            ** Name:(S) LN30    -  LOG entry for finite args only.
2999            **
3000            ** Category:   MATH
3001            **
3002            ** Purpose:
3003            **      To compute  LN(x)
3004            **
3005            ** Entry:
3006            **      Standard floating point math input.
3007            **
3008            ** Exit:
3009            **      Standard floating point math output.
3010            **      XM=1 & P=3 implies  LN(0)
3011            **           & P=4    "     LN(negative)
3012            **
```

```
3013                    **
3014                    ** Calls:      SHF10, (GOES TO DV15?)
3015                    **
3016                    ** Uses.......
3017                    **  Inclusive: P; A,B,C,D; R regs [0]; ST.[10];
3018                    **             HD.ST.[SB,XM];
3019                    **
3020                    ** Stk lvls:   1
3021                    **
3022                    ****************************************************************
3023                    ****************************************************************
3024 0CD7D 7AAD =LN12   GOSUB  splitA
3025
3026 0CD81      =LN15
3027 0CD81 821          XM=0
3028 0CD84 7836         GOSUB  FNPWDS          pull weeds (Also does SB=0)
3029 0CD88 531          GONC   LN30            finite arg.
3030                                           inf. arg. (affine mode)
3031 0CD8B 948   LNINF  ?A=0   S               +inf
3032 0CD8E 00           RTNYES
3033                                           -inf
3034 0CD90 20   =LNNEG  P=     EFIELD
3035 0CD92 3100         LC(2)  =eLOG-          msg code for LN(neg)
3036 0CD96 8C7C  INVNNj GOLONG INVNaN          create NaN & sXCPT:=1
         8F
3037
3038
3039 0CD9C      =LN30                          LN(finite #)
3040 0CD9C 84A          ST=0   10              RESET NEG EXP FLAG
3041 0CD9F AF2          C=0    W               Init. all a(i)=0 & i=0.
3042 0CDA2 AC1          B=0    S
3043 0CDA5 97D          ?B#0   W               X#0?
3044 0CDA8 41           GOYES  LN50            YES
3045                                           X=0
3046 0CDAA 20   LNZR    P=     EFIELD
3047 0CDAC 3100         LC(2)  =eLN0           LN(0) msg code
3048 0CDB0 AC0          A=0    S
3049 0CDB3 BCC          A=-A-1 S               (WANT -inf )
3050 0CDB6 8CC2  LNZR1  GOLONG DZINF
         7F
3051
3052
3053
3054 0CDBC 94C   LN50   ?A#0   S               X<0?
3055 0CDBF 1D           GOYES  LNNEG           YES
3056
3057 0CDC1 D6           C=A    A               C=000...00| n |
3058 0CDC3 8AE          ?C#0   A               n#0?
3059 0CDC6 15           GOYES  LN110
3060
3061
3062
3063
3064
3065
```

```
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075            *.............................*
3076            *   n=0 case.  (1 <= x < 10)  *
3077            *.............................*
3078 0CDC8 2E           P=     14
3079 0CDCA AF9          C=B    W
3080 0CDCD AF7          D=C    W               D=M
3081 0CDD0 A0E          C=C-1  P               FORM MANTISSA OF X-1
3082 0CDD3 97E          ?C#0   W               X#1?
3083 0CDD6 C0           GOYES  LN200
3084
3085            *  x=1  ( ln(1):=0 )   *       SB=0 from FNPWDS in LN15, or
3086                                            from ADDONE in LN1+X.
3087 0CDD8 822          SB=0                   (For other entry points ?)
3088 0CDDB AF1          B=0    W               ANSWER IS 0
3089 0CDDE 6E11         GOTO   LN565           (exit LN15)
3090
3091
3092 0CDE2       LN200
3093 0CDE2 90E          ?C#0   P               NORMALIZE X-1
3094 0CDE5 B0           GOYES  LN210           *
3095 0CDE7 CC           A=A-1  A               *
3096 0CDE9 BF2          CSL    W               *
3097 0CDEC 65FF         GOTO   LN200           *
3098            *  Here A(A)=expon((x-1)/x) & C= its mantissa  *
3099
3100
3101 0CDF0       LN210
3102 0CDF0 714F         GOSUB  LNSUB3          COMPUTE (X-1)/X. Then prepare for
3103                                            DECOMP of 10 - 10*(x-1)/x.
3104                                           .
3105                                           .
3106
3107            ************************************************************
3108            *  Preparation for DECOMP of (10-10*|y|), where Regs (A,B)=y  *
3109            ************************************************************
3110
3111 0CDF4 2E    LN1+X6 P=      14
3112 0CDF6 AF2          C=0    W               Init. all a(i)'s & n' to 0.
3113 0CDF9 108          R0=C                   FOR LN400 CASE (FOR RTN AFTER LNAP)
3114 0CDFC D6           C=A    A               FOR LN300 CASE (C=I&EXP FOR DECOMP)
3115 0CDFE AFA          A=C    W               FOR LN400 CASE (A=EXP FOR LNAP)
3116 0CE01 6D00         GOTO   LN1+X7          A=C=00...00|expon(y)|
3117
3118
3119            *  Set up 10-M and i for entry into middle of DECOMP.  *
3120 0CE05 898   LN1+X8 ?P=    8               Is |y|<10^-7 ? (?)
```

```
3121 0CE08 B7          GOYES  LN400          (use quadratic approx only--LNAP)
3122                                         Note: when P=8 then i=6.
3123 0CE0A 0D          P=P-1
3124 0CE0C B46         C=C+1  S              i=i+1
3125 0CE0F E6    LN1+X7 C=C+1  A              inc. expon
3126 0CE11 53F         GONC   LN1+X8         not done ?
3127
3128 0CE14 471         GOC    LN300          Enter Decomp. C(S)=i, B=10-10y
3129                                         "GOTO"
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143           *..............*
3144           *   n#0 case.  *
3145           *..............*
3146 0CE17      LN110
3147 0CE17 CA          A=A+C  A              EXP<0?
3148 0CE19 570         GONC   LN140          NO
3149 0CE1C FE          C=-C-1 A
3150 0CE1E 85A         ST=1   10             SET NEG EXP FLAG
3151
3152           *                    {   n    if n>0 .      *
3153           *  Here C(A) = n' = {                        *
3154           *                    {  |n|-1  if n<0 .      *
3155
3156                                        .
3157                                        .
3158
3159           ***********************************
3160           *   DECOMPOSITION on (m,n')       *
3161           *     B=m & C(A)=n'               *
3162           ***********************************
3163 0CE21      LN140
3164 0CE21 2E          P=     14
3165 0CE23 B99         B=-B   WP             10 - m
3166 0CE26 A0E         C=C-1  P              a(0):=-1
3167
3168                                         Product Decomp. loops
3169 0CE29 B06   LN310 C=C+1  P              * a(i)=a(i)+1
3170 0CE2C AF4   LN300 A=B    W              * R(i,j)=10 - P(1+10^-i)^j
3171 0CE2F AC7         D=C    S              * D=i
3172 0CE32 6600        GOTO   LN330          *
3173
3174 0CE36 BF5   LN320 BSR    W              * Form R(i,j)*10^-i.
3175 0CE39 A4F   LN330 D=D-1  S              *
```

```
3176 0CE3C 59F          GONC   LN320          *
3177
3178 0CE3F A78          B=A+B  W              * R(i,j)*(1+10^-i)
3179 0CE42 A4D          B=B-1  S              * R(i,j+1)=10^(-i+1)+R(i,j)*
3180 0CE45 53E          GONC   LN310          *                          (1+10^-i)
3181
3182
3183
3184
3185 0CE48 B46          C=C+1  S              * i=i+1  (next p-quot.)
3186 0CE4B AF8          B=A    W              * B=R(i+1,0)=10-P
3187 0CE4E BF1          BSL    W              * Next decade
3188 0CE51 0D           P=P-1                 * Point to next a(i)
3189 0CE53 886          ?P#    6              * More a(i) to determine ?
3190 0CE56 6D           GOYES  LN300          *
3191
3192            *  DECOMP finished  *
3193
3194
3195
3196
3197
3198
3199 0CE58 AFA          A=C    W
3200 0CE5B AF2          C=0    W
3201 0CE5E 20           P=     0
3202 0CE60 3429         LCHEX  99992
           999
3203
3204
3205
3206
3207
3208
3209            *                                                          *
3210            *  At this point:   A=|8|a(0),a(1),...,a(7)|00| n' |       *
3211            *  -------------    B= R                                    *
3212            *                   C=|000...000|99992  (expon (R))        *
3213            *                                                          *
3214            *                                                          *
3215            *  Remaining code composes ln(x) from the DECOMPosition    *
3216            *  of x into a(i)'s and R (in Regs A,B & C).               *
3217            *                                                          *
3218            *                                                          *
3219
3220
3221
3222
3223 0CE67 8AC          ?A#0   A              n'#0 ?
3224 0CE6A C2           GOYES  LN450
3225
3226
3227
3228
3229            *  1<x<10 case -- use quadratic approx (LNAP)   *
```

```
3230 0CE6C 100          R0=A                    save a(i)'s & n'
3231 0CE6F AFE          ACEX   W                A(A)=99992,A(S)=0; C=a(i)'s & n'
3232
3233 0CE72 95E          ?C#0   M                Some a(i)#0 ?
3234 0CE75 B0           GOYES  LN420
3235
3236
3237            *   All a(i)=0   ( x<10^-7 ?)   *
3238 0CE77 CC           A=A-1  A                (A,B)=R UNNORM.(DECR EX not BSR)
3239 0CE79 7080         GOSUB  LN565            NORM. R  (mant in W-field. Need to
3240                                             check for carry out in B(S)--former
3241                                             bug for LN(1-1E-8))
3242                                             "GOSUB MPY150" RTNS WITH CRY. CL.
3243 0CE7D 550          GONC   LN400            "GOTO"
3244
3245
3246            *  Some a(i)#0   *
3247 0CE80 BF5   LN420  BSR    W                R/10
3248
3249            *  Quadratic approx. to -ln(1-R/10)  *
3250 0CE83 7E80  LN400  GOSUB  LNAP             t/(1-t/2), where t=R/10.
3251 0CE87 AF6          C=A    W
3252 0CE8A AF7          D=C    W
3253 0CE8D 110          A=R0                    restore a(i)'s & n'.
3254 0CE90 27           P=     7
3255 0CE92 6010         GOTO   LN431
3256
3257
3258
3259
3260
3261
3262
3263            *  x>=10 and x<=0.1 cases. (n'#0 -- linear approx)  *
3264 0CE96       LN450
3265 0CE96 26           P=     6
3266 0CE98 B04          A=A+1  P                Set n'#0 flag A(6)=1 for P-SUM
3267 0CE9B 27           P=     7
3268 0CE9D AF7          D=C    W                expon of remainder
3269
3270
3271
3272
3273            *.....................................*
3274            *  Add in P-SUM to -ln(1-R/10) approx.  *
3275            *.....................................*
3276 0CEA0 BF5   LN460  BSR    W                PSUEDO-MULTIPLY LOOP
3277 0CEA3 7C32  LN431  GOSUB  LNC20            * Get ln(1+10^-i) const.
3278 0CEA7 7B50         GOSUB  PMUL             * + a(i)*ln(1+10^-i)
3279 0CEAB 958          ?A=0   M                * all a(i) & n' = 0?
3280 0CEAE 63           GOYES  LN530            * Yes. Exit now to avoid
3281                                             *  shifting off significant
3282                                             *  digits of P-SUM - ln(1-R/10)
3283                                             *  for 0 a(i)'s. (Not "full" P-SUM)
3284 0CEB0 88F          ?P#    15               * More a(i)'s to do ?
```

```
3285 OCEB3 DE          GOYES  LN460          *
3286
3287
3288
3289
3290           *  Round [ln(10/m)]  (only when using "full" P-SUM. Why?)
3291 OCEB5 AF2         C=0    W
3292 OCEB8 20          P=     0
3293 OCEBA A89         C=B    P
3294 OCEBD A71         B=B+C  W              ROUND TO 15 DIGITS
3295 OCEC0 BF5         BSR    W
3296
3297
3298
3299           *  B=[ln(10/m)].  (n'#0 cases)
3300 OCEC3 7EC2        GOSUB  LNC10          FETCH LN(10)
3301 OCEC7 87A         ?ST=1  10             NEG EXP FOR X?
3302 OCECA 50          GOYES  LN520           (0<x<0.1 case)
3303
3304 OCECC B7D         B=C-B  W              ln10 - [ln(10/m)]. (x>10 case)
3305
3306
3307
3308           *.....................*
3309           * Add in n'*ln10 term  *
3310           *.....................*
3311 OCECF 7330  LN520  GOSUB  PMUL           Mult. digit of n' and ln10
3312 OCED3 8A8         ?A=0   A              No more digits of n' ?
3313 OCED6 E0          GOYES  LN530          (exit LN15)
3314 OCED8 BF5         BSR    W              next decade of n'
3315 OCEDB 53F         GONC   LN520          "GOTO" next digit multiply
3316
3317
3318
3319
3320
3321           *                                        *
3322           *    LN15 exit -- for all but x=1 case.  *
3323           *    (LN1+X exit --   "       x=0  "  )  *
3324           *                                        *
3325
3326 OCEDE BF5   LN540  BSR    W
3327 OCEE1 490         GOC    LN550          "GOTO"
3328
3329 OCEE4 94D   LN530  ?B#0   S              a carry out ?
3330 OCEE7 7F          GOYES  LN540
3331 OCEE9 CF          D=D-1  A
3332 OCEEB       LN550
3333 OCEEB 7000        GOSUB  =SETSB         set SB=1
3334 OCEEF AFB         C=D    W
3335 OCEF2 AFA         A=C    W
3336 OCEF5 86A         ?ST=0  10             NEG EXP ?
3337 OCEF8 50          GOYES  LN565          NO
3338 OCEFA BCC         A=-A-1 S
3339 OCEFD 8CD7  LN565 GOLONG MPY150         (Changed from SHF10 for GOSUB LN565
```

```
            5F
   3340                                        above LN450 to fix LN(1-1E-8)
   3341
   3342             ******** END LN15 *********
   3343
   3344
   3345
   3346
   3347
   3348
   3349
   3350             ***** SUBROUTINE  PMUL  *******
   3351
   3352 0CF03 A71    PMUL1  B=B+C  W
   3353 0CF06 A0C    PMUL   A=A-1  P
   3354 0CF09 59F           GONC   PMUL1
   3355 0CF0C A80           A=0    P
   3356 0CF0F E7            D=D+1  A
   3357 0CF11 0C            P=P+1
   3358 0CF13 01     PMUL2  RTN
   3359             ********************************
   3360
   3361
   3362
   3363
   3364
   3365
   3366
   3367             ************************************************************
   3368             ***                      LNAP
   3369             ***
   3370             ***      INPUT:  B=MANTISSA OF R
   3371             ***              A(A)= EXPON. OF R, A(S)= sign(R)
   3372             ***      OUTPUT: RETURNS R/(1-|R|/2) IN A & B
   3373             ***
   3374             ***      CONDITION: Must have input satisfying |R|<1 !
   3375             ***
   3376             ***      DESTROYS: A(A),B,C,D,P
   3377             ***      SUBROUTINE LEVELS: 0
   3378             ************************************************************
   3379 0CF15 AF9  =LNAP   C=B    W
   3380 0CF18 AF7          D=C    W
   3381 0CF1B D8           B=A    A
   3382 0CF1D A77          D=D+D  W
   3383 0CF20 A77          D=D+D  W
   3384 0CF23 A73          D=C+D  W
   3385 0CF26 BF7   LNAP1  DSR    W
   3386 0CF29 97B          ?D=0   W
   3387 0CF2C 31           GOYES  LNAP2
   3388 0CF2E E5           B=B+1  A
   3389 0CF30 55F          GONC   LNAP1
   3390 0CF33 BFB          D=-D   W
   3391 0CF36 AC3          D=0    S
   3392 0CF39 E4           A=A+1  A
   3393 0CF3B 69FD         GOTO   LNSUB3
```

```
3394 0CF3F AF5   LNAP2  B=C    W
3395 0CF42 01           RTN
3396            ************************************************************
3397            ***                      EXP
3398            ***
3399            ***       INPUT AND OUTPUT: STANDARD MATH. RETURNS E^X
3400            ***       DESTROYS: A,B,C,D,P,SB,S10,S11,R0
3401            ***       SUBROUTINE LEVELS: 1
3402            ***
3403            ***
3404            ***                      EX-1
3405            ***
3406            ***       INPUT: STANDARD MATH
3407            ***       OUTPUT: RETURNS E^X IN A&B.
3408            ***               RETURNS (E^X)-1 IN R0,R1
3409            ***       DESTROYS: A,B,C,D,P,SB,S10,S11,R0,R1 ******
3410            ***       SUBROUTINE LEVELS: 1
3411            ************************************************************
3412            *
3413            ********************************************************************
3414            ********************************************************************
3415            **
3416            ** Name:(S) EXP15   -  EXP(x)    (exponential fcn)
3417            ** Name:(S) EX-115  -  EXP(x)-1  (EXPM1(x))
3418            ** Name:(S) DXP100  -  EXP for double precision arg
3419            **
3420            ** Category:   MATH
3421            **
3422            ** Purpose:
3423            **      To compute  e^x
3424            **
3425            ** Entry:
3426            **      Standard floating point math input.
3427            **      ( Uses s11 to distinguish e^x from [e^x - 1] )
3428            **      DXP100: finite args only; s11=0; R0=low order
3429            **       digits of double precision argument.
3430            **
3431            ** Exit:
3432            **      Standard floating point math output for EXP15.
3433            **      EX-115 outputs e^x in (A,B)&SB and [e^x - 1] in
3434            **       (R1,R0)&SB  (SB is the same for both).
3435            **
3436            ** Calls:   DBLSUB,SHFLAC,1/X15S,STAB1,EXAB1,ADDONE,SUBONE
3437            **          and other local subroutines.
3438            **
3439            ** Uses.......
3440            **  Inclusive: P; A,B,C,D; R regs [0]; ST.[10,11];
3441            **                HD.ST.[SB,XM];
3442            **             EX-115 also uses R1.
3443            **
3444            **
3445            ** Stk lvls:   1
3446            **
3447            ** NOTE:
3448            **      When xpon(e^x)>19999 then EXP15(x):=9.99...99E+19999 .
```

```
3449            **      When     "    < -19999 then EXP15(x):=9.9..99E-19999 .
3450            **
3451            **
3452            ************************************************************************
3453            ************************************************************************
3454 0CF44 73EB =EX-112 GOSUB  splitA
3455
3456 0CF48      =EX-115
3457 0CF48 821          XM=0
3458 0CF4B 85B          ST=1   11              SET E^X-1 FLAG
3459 0CF4E 7784         GOSUB  STAB1           (For NaN input only)
3460 0CF52 6D00         GOTO   EXP90
3461
3462 0CF56 71DB =EXP12  GOSUB  splitA
3463 0CF5A      =EXP15
3464 0CF5A 821  =EXP15M XM=0
3465 0CF5D 84B          ST=0   11              INIT e^x - 1 flag to false.
3466 0CF60 7C54 =EXP90  GOSUB  FNPWDS          (Also does SB=0)
3467 0CF64 541          GONC   EXP100
3468
3469            *   Inf  argument   *
3470 0CF67 948          ?A=0   S               +Inf ?
3471 0CF6A 80           GOYES  exp710          EXP(+Inf) = +Inf
3472
3473            *  -Inf  argument   *
3474 0CF6C AF0          A=0    W
3475 0CF6F AF1          B=0    W
3476 0CF72 6E92 exp710  GOTO   EXP710          EXP(-Inf) = 0
3477
3478
3479
3480 0CF76      =EXP16M                        assumes 16 digit normal finite no.
3481                                            NO NaN,inf or denorm handling !!!
3482                                            input: s11=1 for [Y^X-1]
3483 0CF76 821          XM=0
3484
3485 0CF79      =EXP100
3486 0CF79 AF2          C=0    W
3487 0CF7C 108          R0=C                   (low digits of dbl x )
3488 0CF7F      =DXP100                        DBL EXP (R0= LOW DIGITS)
3489 0CF7F 85A          ST=1   10              INIT neg mant flag
3490 0CF82 94C          ?A#0   S               X NEG?
3491 0CF85 50           GOYES  EXP110          YES
3492 0CF87 84A          ST=0   10              RESET NEG MANT FLAG
3493
3494 0CF8A AC0   EXP110 A=0    S
3495 0CF8D AF3          D=0    W               clear p-quot&exp ans (for neg expn)
3496 0CF90 D6    EXP115 C=A    A
3497 0CF92 AC2          C=0    S               ????? needed for neg expn case??
3498 0CF95 C6           C=C+C  A               NEG EXP?
3499 0CF97 582          GONC   DXP200          NO  ( n>=0)
3500
3501 0CF9A D6           C=A    A
3502 0CF9C 949          ?B=0   S               16 DIGIT MANTISSA?
3503 0CF9F B0           GOYES  EXP120          NO
```

```
3504
3505            *   16 digit mantissa    *
3506 0CFA1 E4          A=A+1  A
3507 0CFA3 BF5         BSR    W
3508 0CFA6 69EF        GOTO   EXP115
3509
3510
3511
3512
3513            *   |x| < 1  case, so          *
3514            *  prepare for DECOMPOSITION;  *
3515            *   determine how small |x| is *
3516
3517 0CFAA 2F    EXP120 P=     15
3518 0CFAC 0D    EXP130 P=P-1
3519 0CFAE 896         ?P=    6              |x| < 1E-8 ?
3520 0CFB1 B0          GOYES  EXP131
3521 0CFB3 E6          C=C+1  A
3522 0CFB5 56F         GONC   EXP130
3523 0CFB8 6E70        GOTO   EXP400         enter DECOMP.
3524 0CFBC 6180  EXP131 GOTO   EXP500
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535            **********************
3536            *   |X| >= 1  case   *   Double Reduction by ln10 (x'=x-n'*ln10)
3537            **********************
3538
3539 0CFC0 120   DXP200 AROEX                 R0=0...XPON(X); A=X_low
3540 0CFC3 AFC         ABEX   W               A=X_high; B=X_low
3541
3542            *   Extra Precision ln10    *
3543 0CFC6 AF2         C=0    W
3544 0CFC9 2C          P=     12
3545 0CFCB 3348        LCHEX  5684           (low digits = 5684018- )
           65
3546 0CFD1 AF7         D=C    W
3547 0CFD4 7AB1        GOSUB  LNC10+         + LN10
3548 0CFD8 CE          C=C-1  A              (C,D)=2.3025 85092 99404 / 56840..0
3549 0CFDA AFD         BCEX   W
3550 0CFDD D2          C=0    A
3551                                         A=...X_high...
3552                                         B=0230258509299404
3553                                         C=...X_low..00000
3554                                         D=568400....00000
3555
3556                                         R0=0--------xponX
3557 0CFDF 25          P=     5
```

```
3558 0CFE1 CE           C=C-1  A
3559
3560            *  Perform a double precision reduction by ln10  *
3561 0CFE3 E6    DXP210 C=C+1  A              COMPUTE XPON ANS
3562 0CFE5 8E00  DXP220 GOSUBL =DBLSUB        x'-ln10
           00
3563 0CFEB 57F          GONC   DXP210
3564 0CFEE 8E00         GOSUBL =SHFLAC        DBL SHIFT
           00
3565 0CFF4 120          AROEX
3566 0CFF7 CC           A=A-1  A              DEC XPON(X)
3567 0CFF9 120          AROEX
3568 0CFFC 4B0          GOC    DXP225
3569 0CFFF 90A          ?C=0   P              EXPON ANS <=99999?
3570 0D002 3E           GOYES  DXP220
3571
3572 0D004 62F1         GOTO   OVF-15         (OV/UN occured)
3573
3574
3575
3576 0D008 B96   DXP225 CSR    WP             shift expon ans back
3577 0D00B AF8          B=A    W              B=X_high
3578 0D00E 110          A=R0                  A=0...99999 ?
3579 0D011 AF3          D=0    W              (zero for p-quots)
3580 0D014 D7           D=C    A              D(A)=expon ans
3581
3582
3583
3584 0D016 77C1         GOSUB  OVF15?         check OVF/UNF
3585 0D01A 5C1          GONC   EXP400         no internal exception
3586 0D01D 01           RTN
3587
3588            ********** OVF/UNF exit of EXP *****************
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600            ********************
3601            *   DECOMPOSE x'   *
3602            ********************
3603                                          PSUEDO-DIVIDE LOOP
3604 0D01F B07   EXP410 D=D+1  P              * build a(i)
3605 0D022 B71   EXP420 B=B-C  W              * R(i) = R(i) - ln(1+10^-i)
3606 0D025 59F          GONC   EXP410         *
3607                                          *
3608 0D028 A71          B=B+C  W              * R(i)
3609 0D02B 897          ?P=    7              *
3610 0D02E 01           GOYES  EXP500         *
```

```
3611                                          *
3612 0D030 BF1          BSL    W              * next decade
3613 0D033 CC           A=A-1  A              * exponent of remainder
3614 0D035 0D           P=P-1                 *
3615 0D037 78A0 EXP400  GOSUB  LNC20          * FETCH LOG CONSTANT
3616 0D03B 46E          GOC    EXP420         * "GOTO" (LNC20 HAS RTNSC)
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627            ******************************
3628            *    Compose  [e^x' - 1]     *
3629            ******************************
3630
3631 0D03E        EXP500
3632 0D03E 86B          ?ST=0  11             E^X-1?
3633 0D041 21           GOYES  EXP565         NO
3634 0D043 AFB          C=D    W
3635 0D046 108          R0=C                  SAVE PQUOT AND EXP ANS
3636 0D049 78CE         GOSUB  LNAP           COMPUTE R/(1-R/2)
3637                                          Quadratic approx. of e^R - 1
3638 0D04D 118          C=R0
3639 0D050 AF7          D=C    W              RESTORE PQUOT AND EXP ANS
3640
3641
3642 0D053 D6   EXP565  C=A    A
3643 0D055 2F   EXP570  P=     15
3644 0D057 307          LCHEX  7              i=7
3645 0D05A 26           P=     6
3646 0D05C 8AB          ?D=0   A              ZERO EXP?
3647 0D05F 50           GOYES  EXP550         YES
3648 0D061 B07          D=D+1  P              SET D(6)=1 (flags n'#0 ).
3649
3650
3651
3652
3653            *   Start Composing [e^x - 1]   *
3654
3655            *                            FORM PRODUCT
3656 0D064 95B  EXP550 ?D=0   M               PQUOT AND EXP ANSWER = 0 ?
3657 0D067 F4           GOYES  EXP600         YES  (|x| < ln2)
3658 0D069 0C           P=P+1
3659 0D06B 90B  EXP560 ?D=0   P
3660 0D06E 42           GOYES  EXP520
3661 0D070 A0F          D=D-1  P              won't carry since D#0 P .
3662 0D073 AF4          A=B    W
3663 0D076 AC7          D=C    S              D(S)=i
3664 0D079 550          GONC   EXP530         "GOTO"
3665
```

```
3666
3667 0D07C BF4   EXP540 ASR    W              form [P(1+10^-i)^j - 10]*10^-i
3668 0D07F A4E   EXP530 C=C-1  S
3669 0D082 59F          GONC   EXP540
3670
3671 0D085 A78          B=A+B  W              [ " ](1+10^-i)
3672 0D088 B45          B=B+1  S              ... + 10^(-i+1)
3673 0D08B ACB          C=D    S              i
3674 0D08E 6CDF         GOTO   EXP560
3675
3676 0D092 E6    EXP520 C=C+1  A
3677 0D094 BF5          BSR    W
3678 0D097 A4E          C=C-1  S
3679 0D09A 59C          GONC   EXP550
3680
3681
3682
3683
3684            *  End Composition   *
3685            *  |x| >= ln2 case   *
3686            *  [e^x' - 1]        *
3687 0D09D DB           C=D    A              expon ans = n'
3688 0D09F B05          B=B+1  P              mant e^x'
3689 0D0A2 DA           A=C    A
3690 0D0A4 AC0          A=0    S
3691                                          Here (A,B)=e^|x|
3692 0D0A7 86A          ?ST=0  10             NEGATIVE MANTISSA?
3693 0D0AA 80           GOYES  EXP521         NO
3694 0D0AC 8E29         GOSUBL 1/X15S
         2F
3695 0D0B2 6E51  EXP521 GOTO   EXP710         e^x
3696
3697            *   EXIT EXP    *
3698            *---------------*
3699
3700
3701
3702
3703
3704
3705
3706            *  All remaining p-quotients are 0, and expon(ans)=0 *
3707            *   i.e.  |x| < ln2   <=>   0 <= [e^|x| - 1] < 1 .
3708 0D0B6 AC0   EXP600 A=0    S
3709 0D0B9 DA           A=C    A
3710 0D0BB 86A          ?ST=0  10             NEGATIVE MANTISSA?
3711 0D0BE 90           GOYES  EXP740         NO
3712
3713            *    x<0  case     *
3714 0D0C0 BCC          A=-A-1 S              y = 1-e^|x|
3715 0D0C3 745C         GOSUB  LNSUB          y/(1+|y|) = e^x - 1
3716
3717            *   have  [e^x - 1].   *
3718 0D0C7 86B   EXP740 ?ST=0  11             E^X-1 WANTED?
3719 0D0CA 60           GOYES  EXP750         NO
```

```
3720
3721 0D0CC 7903          GOSUB  STAB1          STORE E^X-1 IN MATH SCRATCH
3722
3723 0D0D0        EXP750
3724 0D0D0 979           ?B=0   W              [e^x - 1] = 0 ?
3725 0D0D3 A0            GOYES  EXP760         Yes (0+1 will clear SB on exit)
3726
3727 0D0D5 7400          GOSUB  EXP760         e^x  (ADDONE may clear SB !)
3728 0D0D9 6000          GOTO   =SETSB         SB:=1 since x#0; exit EXP.
3729
3730 0D0DD 8C15  EXP760  GOLONG ADDONE         COMPUTE e^x. ([e^x-1]+1 # 0 so
             2F
3731                                               ignore rnd'g bits input)
3732             *-------------------------------------------------------------*
3733
3734
3735
3736
3737
3738
3739 0D0E3        LNC20                         RTNS WITH CARRY SET.
3740 0D0E3 AF2             C=0    W
3741 0D0E6 89E           ?P=    14
3742 0D0E9 D3            GOYES  LNC30
3743 0D0EB A5E           C=C-1  M
3744 0D0EE 304           LCHEX  4
3745 0D0F1 B56           C=C+1  M
3746 0D0F4 89D           ?P=    13
3747 0D0F7 64            GOYES  LNC40
3748 0D0F9 89C           ?P=    12
3749 0D0FC 65            GOYES  LNC50
3750 0D0FE 89B           ?P=    11
3751 0D101 46            GOYES  LNC60
3752 0D103 89A           ?P=    10
3753 0D106 07            GOYES  LNC70
3754 0D108 899           ?P=    9
3755 0D10B A7            GOYES  LNC80
3756 0D10D 897           ?P=    7
3757 0D110 D0            GOYES  LNC90
3758 0D112 20            P=     0
3759 0D114 3233          LCHEX  333
             3
3760 0D119 28            P=     8
3761 0D11B 02            RTNSC
3762 0D11D 20    LNC90   P=     0
3763 0D11F 303           LCHEX  3
3764 0D122 27            P=     7
3765 0D124 01            RTN
3766 0D126 20    LNC30   P=     0
3767 0D128 3E54          LCHEX  693147180559945
             9955
             0817
             4139
             6
3768 0D139 2E            P=     14
```

```
3769 0D13B 01           RTN
3770 0D13D 20    LNC40  P=     0
3771 0D13F 3C94         LCHEX  3101798043249
           2340
           8971
           013
3772 0D14E 2D           P=     13
3773 0D150 01           RTN
3774 0D152 20    LNC50  P=     0
3775 0D154 3A80         LCHEX  33085316808
           8613
           5803
           3
3776 0D161 2C           P=     12
3777 0D163 01           RTN
3778 0D165 20    LNC60  P=     0
3779 0D167 3833         LCHEX  333083533
           5380
           333
3780 0D172 2B           P=     11
3781 0D174 01           RTN
3782 0D176 20    LNC70  P=     0
3783 0D178 3638         LCHEX  3333083
           0333
           3
3784 0D181 2A           P=     10
3785 0D183 01           RTN
3786 0D185 20    LNC80  P=     0
3787 0D187 3433         LCHEX  33333
           333
3788 0D18E 29           P=     9
3789 0D190 01           RTN
3790
3791 0D192 AC2  LNC10+  C=0    S              FOR +ln10
3792 0D195 20   =LNC10  P=     0
3793 0D197 3E50         LCHEX  230258509299405
           4992
           9058
           5203
           2
3794 0D1A8 01           RTN
3795
3796
3797
3798          ************************************************************************
3799          ************************************************************************
3800          **
3801          ** Name:(S) LGT15   -  Log base 10
3802          **
3803          ** Category:   MATH
3804          **
3805          ** Purpose:
3806          **      To compute the base 10 logarithm of x.
3807          **
3808          ** Entry:
```

```
3809            **      Standard floating point math input.
3810            **
3811            ** Exit:
3812            **      Standard floating point math output.
3813            **
3814            ** Calls:      NRMLAB, EX15, LN15, LNC10+, DV15S, MAKE1
3815            **
3816            ** Uses.......
3817            **  Inclusive: P; A,B,C,D; R regs [0]; ST.[10];
3818            **                HD.ST.[SB,XM];
3819            **
3820            ** Stk lvls:   2
3821            **
3822            ** NOTE:
3823            **      LGT(10^n) returns n exactly.
3824            **
3825            ********************************************************************
3826            ********************************************************************
3827
3828 0D1AA 7D79 =LGT12  GOSUB  splitA
3829 0D1AE      =LGT15
3830 0D1AE 8E12         GOSUBL NRMLAB          This will make Inf & NaN mant#1
          5F
3831 0D1B4 94C          ?A#0   S               A<0 (or =-0 )
3832 0D1B7 F0           GOYES  LGT50           use usual code
3833
3834            * check for LGT(10^n)  *
3835 0D1B9 8E00         GOSUBL =MAKE1          C:=0100...00 & compare with B
          00
3836 0D1BF 560          GONC   LGT50           Mant#1
3837 0D1C2 6000         GOTO   =EX15M          rtn expon(x) exactly.
3838
3839
3840
3841 0D1C6 77BB LGT50   GOSUB  LN15            COMPUTE LN(X)
3842 0D1CA 831          ?XM=0                  No LN xcpt'n occurred?
3843 0D1CD 40           GOYES  XLOG20
3844 0D1CF 01           RTN                    LGT(x) = LN(x) if x=0 or x<0.
3845                                             Same error msg., too.
3846
3847
3848            *  usual code   *
3849 0D1D1 7DBF XLOG20  GOSUB  LNC10+          LN10
3850 0D1D5 AF3          D=0    W
3851 0D1D8 AFF          CDEX   W
3852 0D1DB 8C5D         GOLONG DV15S           LN(X)/LN10. SB=0 here, only if
          2F
3853                                            x = Inf or NaN (x=1 already trap-
3854                                            ped out).
3855
3856
3857            *********  END LGT15  **********
3858
3859
3860
```

```
3861
3862
3863
3864
3865
3866
3867
3868            *----------------------------*
3869
3870 0D1E1      =OVF15?                       input: D(A)=|expon|
3871                                               s10=sign of expon
3872                                               s11=will put rstl-1 into R0&1
3873                                          output: carry clr ==> no xcpt
3874                                                    uses only P:=14.
3875                                                  carry set ==> xcpt. and
3876                                                    (A,B):=9.99..E(+/-)19999.
3877                                                    uses ABC&D,P.
3878
3879 0D1E1 24           P=     4
3880 0D1E3 90B          ?D=0   P              EXP ANS <=9999?
3881 0D1E6 D0           GOYES  EXP240         YES
3882 0D1E8 A0F          D=D-1  P
3883 0D1EB 90F          ?D#0   P              EXP ANS >19999?
3884 0D1EE 90           GOYES  OVF-15
3885 0D1F0 B07          D=D+1  P
3886 0D1F3 2E    EXP240 P=     14
3887 0D1F5 03           RTNCC                 no internal xcpt
3888
3889
3890 0D1F7 AF2  =OVF-15 C=0    W              OVER-UNDERFLOW. create reslt.
3891 0D1FA 2E           P=     14
3892 0D1FC A1E          C=C-1  WP
3893 0D1FF AF5          B=C    W
3894 0D202 24           P=     4
3895 0D204 301          LCHEX  1              C(A)= 19999  (for OVFL)
3896 0D207 86A          ?ST=0  10             NEGATIVE MANTISSA FLAG SET?
3897 0D20A 40           GOYES  EXP700         NO
3898 0D20C FA           C=-C   A              C(A)=80001  (-19999 for UNDFL)
3899 0D20E AFA   EXP700 A=C    W
3900
3901
3902 0D211      EXP710
3903 0D211 86B          ?ST=0  11             E^X-1 NOT SET?
3904 0D214 00           RTNYES
3905
3906
3907
3908 0D216 7FB1         GOSUB  STAB1          STORE E^X
3909 0D21A 8E70         GOSUBL SUBONE         COMPUTE e^x-1 (#0 here, would be
          1F
3910                                           inacc,so ignore rnd'g bits input)
3911 0D220 73C1         GOSUB  EXAB1          EXCHANGE E^X AND E^X-1
3912 0D224 02           RTNSC
3913            *----------------------*
3914
```

```
3915
3916
3917
3918
3919
3920
3921
3922
3923            *************************
3924            *        PWRTEN         *
3925            *************************
3926
3927 0D226      PWRTEN                          special code for (1x10^k)^N.
3928                                            INPUT: (A,B)=Y, (C,D)=N.
3929                                            Y,N
3930 0D226 7000          GOSUB  =EX15M
3931 0D22A 8EA0          GOSUBL =MP2-15         kN    in +/-{0,1,2,...}
           2F
3932 0D230 84A           ST=0   10              S10=SGN OF kN
3933 0D233 948           ?A=0   S
3934 0D236 50            GOYES  PWR40
3935 0D238 85A           ST=1   10
3936 0D23B D2    PWR40   C=0    A
3937 0D23D 20            P=     0
3938 0D23F 304           LCHEX  4
3939 0D242 8B6           ?A>C   A               expon(kN)>4  (i.e. >5 digits)
3940 0D245 2B            GOYES  OVF-15          OVF/UNF create 9.9..E(+/-)19999
3941                                              according to s10.
3942
3943 0D247 811           BSLC                   * Shift kN into expon field.
3944 0D24A 811   PWR60   BSLC                   *
3945 0D24D CC            A=A-1  A               *
3946 0D24F 5AF           GONC   PWR60           *
3947
3948 0D252 D9            C=B    A
3949 0D254 D7            D=C    A               D(A):=ans expon (|kN|) for OVF15? .
3950 0D256 778F          GOSUB  OVF15?          OVF/UNF ?   (looks at s10)
3951 0D25A 400           RTNC                   yes. rtn 9.99..E(+/-)19999.
3952
3953
3954
3955 0D25D DB            C=D    A
3956 0D25F DA            A=C    A               restore legal expon. A(A):=|kN|
3957 0D261 86A           ?ST=0  10              Sgn(kN)="+" ?
3958 0D264 40            GOYES  PWR80
3959 0D266 F8            A=-A   A               kN
3960
3961 0D268 AC0   PWR80   A=0    S               10^kN is non-neg.
3962 0D26B AF1           B=0    W               Make Mantissa = 1.
3963 0D26E B05           B=B+1  P                (P=14 FROM OVF15? )
3964 0D271 5F9           GONC   EXP710          "GOTO"
3965            *----------------------*
3966
3967
3968            ************************************************************
```

```
3969            ****************************************************************
3970            **
3971            ** Name:(S) YX2-15  -  Y to the X power
3972            ** Name:(S) YX2-12  -  Y^X for 12-form arguments
3973            **
3974            ** Category:   MATH
3975            **
3976            ** Purpose:
3977            **      To compute  y^x
3978            **
3979            ** Entry:
3980            **      Standard floating point math input.
3981            **      s11 can be used to compute [Y^X - 1] by entering
3982            **         later with s11=1.
3983            **
3984            ** Exit:
3985            **      Standard floating point math output.
3986            **
3987            ** Calls:      LN   , EXP
3988            **
3989            ** Uses.......
3990            **  Inclusive: P; A,B,C,D; R regs [0,2,3]; ST.[sY=INF(s8),10,11];
3991            **                HD.ST.[SB,XM];
3992            **             [y^x - 1] uses R1 also.
3993            **
3994            ** Stk lvls:   3
3995            **
3996            ** NOTE:
3997            **      If |y^x| > 1E20000 or <1E-20000 then y^x-->1E(+/-)20000,
3998            **        these are the internal ovf/unf thresholds.
3999            **
4000            ****************************************************************
4001            ****************************************************************
4002 0D274      =YX2-12
4003 0D274 8EAB         GOSUBL SPLTAC
           6F
4004 0D27A      =YX2-15
4005 0D27A 821          XM=0
4006 0D27D 822          SB=0
4007 0D280 7000         GOSUB  =MSN15          ELIMINATE NANS
4008 0D284 400          RTNC                   return with correct NaN
4009 0D287 84B          ST=0   11
4010 0D28A 848          ST=0   sY=INF          init y#inf
4011 0D28D AC1          B=0    S               CLEAR ANY JUNK
4012 0D290 AC3          D=0    S
4013 0D293 8EC3         GOSUBL NRMLAB
           4F
4014 0D299 550          GONC   YX30            finite  y
4015
4016            *  y=inf    RECALL: inf has mant#0 & expon=00F00 from NRMLAB
4017 0D29C 858          ST=1   sY=INF
4018
4019
4020 0D29F       YX30
4021 0D29F 8E1B         GOSUBL NRMLCD
```

```
                6F
4022
4023                 *  NOTE: for x=inf NRMLCD yields: mant#0 & expon=00F00
4024
4025
4026                                                    finite & +/- inf cases
4027
4028 0D2A5 7E71              GOSUB  STCD2               STORE X AWAY. Y STILL IN A&B
4029 0D2A9 948               ?A=0   S                   y >= +0 ?
4030 0D2AC 60                GOYES  YX050               YES
4031
4032 0D2AE 6880              GOTO   YX200               (y <= -0)
4033
4034 0D2B2 97D  =YX050       ?B#0   W                   Y#0?
4035 0D2B5 60                GOYES  YX055               YES
4036 0D2B7 62E0              GOTO   YX700               (y=+0)
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049                 ********************
4050                 *   Y > +0  case   *
4051                 ********************
4052
4053 0D2BB            YX055                             Y,X  R2&3 = X
4054                 * test for mant(Y)=1 *
4055 0D2BB 8E00              GOSUBL =MAKE1              C:=0100...00 & is compared with B.
                00
4056 0D2C1 5C2               GONC   YX056               Mant#1
4057
4058
4059
4060                 *   Mant(y) = 1    *
4061
4062                 * if Y is non-integer or X=Inf then goto usual code  *
4063 0D2C4 7451              GOSUB  RCCD2               Y,X    R2&3=X
4064 0D2C8 734A              GOSUB  FINITC              x<Inf ?
4065 0D2CC 412               GOC    YX056               send (1x10^k)^Inf thru usual code.
4066 0D2CF 8E2C              GOSUBL XYEX                X,Y
                3F
4067 0D2D5 D7                D=C    A                   save
4068 0D2D7 8E71              GOSUBL CLRFRC              N,Y'  (carry set if integer)
                4F
4069 0D2DD DB                C=D    A
4070 0D2DF D3                D=0    A                   restore 1 mant.
4071 0D2E1 8E0B              GOSUBL XYEX                Y,N  (preserves carry)
                3F
```

```
4072 0D2E7 560          GONC   YX056          X is non-int, goto usual code.
4073 0D2EA 6B3F         GOTO   PWRTEN         use special (1x10^k)^N algor.(gives
4074                                            exact 10^499,10^-499, etc.)
4075
4076
4077
4078
4079            *   Mant(y) # 1    or    x non-int     *
4080
4081            *     here (A,B)=y and SCR2=x     *
4082 0D2EE      YX056                          usual e^(x*ln y) calculation.
4083
4084 0D2EE 7F8A         GOSUB  LN15           COMPUTE LN(Y)  (no xcpt will occur)
4085 0D2F2 7621         GOSUB  RCCD2          FETCH X TO C&D
4086
4087 0D2F6 8EE3         GOSUBL MP2-15         COMPUTE X*LN(Y)
           1F
4088                                          (XM set properly)
4089 0D2FC 831          ?XM=0                 no xcpt'n?
4090 0D2FF 32           GOYES  YX060
4091
4092            *     exception on x*lny      *
4093 0D301 20           P=     EFIELD
4094 0D303 D2           C=0    A
4095 0D305 3100         LC(2)  =eINF^0
4096 0D309 878          ?ST=1  sY=INF         y=inf?********************
4097 0D30C 90           GOYES  ONE15          ******* inf^0 err *******
4098                                          **************************
4099
4100                                          **************************
4101 0D30E 3100         LC(2)  =e1^INF        ******* 1^inf err *******
4102                                          **************************
4103
4104 0D312 5E3          GONC   Ivn            "goto" INVNaN
4105
4106 0D315 AF0  ONE15   A=0    W
4107 0D318 AF1          B=0    W
4108 0D31B 2E           P=     14             P=TYP0^0 EXCEPTION
4109 0D31D B05          B=B+1  P              1
4110                                          (gosub EXP710; P=14. will correct
4111                                            this case for [Y^X-1] BUT still
4112                                            the msg code in C(A) is lost)
4113
4114 0D320 00           RTNSXM                XM=1 for uRES12
4115
4116
4117
4118
4119            *   Computed x*ln(y) with no exceptions   *
4120
4121 0D322 94F  =YX060  ?D#0   S              16 DIGIT RESULT?
4122 0D325 60           GOYES  YX100
4123 0D327 683C         GOTO   EXP90          EXP. (does not alter s11,
4124                                            allows for [Y^X - 1] calc'n.
4125            *******  EXIT  *********
```

```
4126
4127
4128
4129
4130
4131            *   16 digit result of x*ln(y)   *
4132 0D32B CC    YX100  A=A-1  A              DEC EXP
4133 0D32D AFB          C=D    W
4134 0D330 AFD          BCEX   W              USE 16 DIGIT MANTISSA
4135 0D333 624C         GOTO   EXP16M         COMPUTE E^(X*LN(Y))
4136
4137            ******  EXIT  **********
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151            ******************
4152            *    Y =< -0     *
4153            ******************
4154
4155 0D337 73D0  YX200  GOSUB  EXAB2          SCR=Y, A&B = X
4156 0D33B D6           C=A    A
4157 0D33D C6           C=C+C  A              ABS(X) < 1 ?
4158 0D33F 551          GONC   YX400          !X!>=1 or =0 (non-neg expon)
4159
4160            *   0<|x|<1   *
4161
4162 0D342 78C0 YX250   GOSUB  EXAB2          (A,B)=Y, (R2,3)=X
4163 0D346 979          ?B=0   W              Y=0? (tests for Y=-0 here)
4164 0D349 15           GOYES  YX700          -0^nonint: rtn -0 or -Inf
4165
4166 0D34B       NONINT
4167 0D34B 20           P=     EFIELD
4168                                          ****************************
4169 0D34D 3100         LC(2)  =eNEG^X        ****** neg^(non-int) *******
4170                                          ****************************
4171 0D351 644A  Ivn    GOTO   INVNNj         (INVNaN)
4172
4173
4174
4175
4176 0D355 8E2E  YX400  GOSUBL INFR15         x, SCR=y
           3F
4177 0D35B AC3          D=0    S              init to "even" for >15 dig #
4178 0D35E 88F          ?P#    15             < 15 digits
4179 0D361 A0           GOYES  YX450
```

```
4180
4181              *   P=15   *
4182 0D363 8AE           ?C#0   A              >15 DIGITS (must be even)
4183 0D366 B1            GOYES  YX550
4184 0D368 570           GONC   YX500          =15 DIGITS
4185
4186
4187
4188
4189 0D36B 91D  YX450    ?B#0   WP             Is X non-integer ?
4190 0D36E 4D            GOYES  YX250
4191
4192
4193
4194              *   Test N (int. x) for even/odd   *
4195 0D370 0C     YX500  P=P+1                  POINT TO LSD INTEGER PART OF X
4196 0D372 A89           C=B    P
4197 0D375 A06           C=C+C  P
4198 0D378 A06           C=C+C  P
4199 0D37B A09           C=B+C  P              LSD MOD 2
4200 0D37E A87           D=C    P              D(P):= N mod 2
4201
4202 0D381 7980  YX550   GOSUB  EXAB2          A&B = Y,  SCR=N
4203 0D385 AC0           A=0    S              |Y|
4204 0D388 90B           ?D=0   P              X EVEN?
4205 0D38B B0            GOYES  YX600          YES. Compute |y|^N
4206 0D38D 712F          GOSUB  YX050          Compute |y|^X
4207 0D391 BCC           A=-A-1 S              -|y|^N
4208 0D394 01            RTN
4209
4210
4211 0D396 6B1F  YX600   GOTO   YX050          COMPUTE Y^X
4212
4213              ******** EXITS for neg^integer *****************
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225              ********************
4226              *   Y = 0  case    *
4227              ********************
4228
4229 0D39A 7E70  YX700   GOSUB  RCCD2          Y=+/- 0. RECALL X IN C&D
4230 0D39E 20            P=     EFIELD
4231 0D3A0 97F           ?D#0   W              X#0?
4232 0D3A3 C0            GOYES  YX750
4233 0D3A5 D2            C=0    A
4234                                           **********************
```

```
4235 0D3A7 3100         LC(2)  =e0^0          ****** 0^0 ERR  ******
4236                                          **********************
4237
4238 0D3AB 696F         GOTO   ONE15          "goto"
4239
4240 0D3AF 94A  YX750   ?C=0   S              X NON-NEG? (A,B)=y=+0
4241 0D3B2 A0           GOYES  YX800          YES
4242
4243                                          ************************
4244 0D3B4 3100         LC(2)  =e0^NEG        ****** 0^neg err ******
4245                                          ************************
4246
4247 0D3B8 6DF9         GOTO   LNZR1          +/- Inf  (golong DZINF)
4248
4249 0D3BC 645E  YX800  GOTO   EXP710         ANSWER IS 0
4250
4251
4252
4253
4254
4255
4256
4257            ********************************************************************
4258            ********************************************************************
4259            **
4260            ** Name:(S) FNPWDS  -  Weed out NaNs and Infs
4261            **
4262            ** Category:   MTHUTL
4263            **
4264            ** Purpose:
4265            **      To handle NaN and Inf as arguments to functions.
4266            **
4267            **
4268            ** Entry:
4269            **      AB=x
4270            **
4271            ** Exit:
4272            **      If x is
4273            **              1) finite   ==> RTNCC
4274            **              2) inf      ==> RTNSC
4275            **              3) NaN      ==> abort call'g fn (C=RSTK)
4276            **                              RTN with x (input NaN)
4277            **      DEC Mode
4278            **
4279            **
4280            ** Calls:      FINITA
4281            **
4282            ** Uses.......
4283            **  Inclusive: C(A)
4284            **
4285            ** Stk lvls:   0    (Uses C(A) to save the level.)
4286            **
4287            ********************************************************************
4288            ********************************************************************
4289 0D3C0 822  =FNPWDS SB=0                  init exact.
```

```
4290 0D3C3 07          C=RSTK                  keep 1 subr lev
4291 0D3C5 7A39        GOSUB  FINITA           sets carry for xcpt'l arg
4292 0D3C9 460         GOC    XCPARG
4293 0D3CC 06          RSTK=C                  replace FN addr
4294 0D3CE 03          RTNCC                   CC FOR FINITE ARG
4295
4296 0D3D0 96C   XCPARG ?A#0   B               NaN?
4297 0D3D3 00          RTNYES
4298           *   NaN input--RSTK popped. RTN from calling fcn with NaN  *
4299
4300
4301
4302           *  Inf  *
4303 0D3D5 06          RSTK=C
4304 0D3D7 02          RTNSC
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315           ************************************************************
4316           ***                      CPU SCRATCH                     ***
4317           ***      R0,R1= SCRATCH1,   R0(S)=S, R0(A)=EXP,R1(14-0)=MANT***
4318           ***      R2,R2= SCRATCH2, R2(S)=S, R2(A)=EXP, R3(14-0)=MANT ***
4319           ***                                                      ***
4320           ***      STAB1: COPY A&B IN SCR1. DESTROYS R0,R1 ONLY    ***
4321           ***      RCAB1: RECALL SCR1 TO A&B. DESTROYS A&B ONLY    ***
4322           ***      EXAB1: EXCHANGE A&B WITH SCR1. DESTROYS NOTHING ***
4323           ***                                                      ***
4324           ***      ETC.                                            ***
4325           ***                                                      ***
4326           ***                                                      ***
4327           ************************************************************
4328
4329           ************************************************************
4330           ************************************************************
4331           **
4332           ** Name:(S) STAB1   -  Store AB into scratch 1
4333           ** Name:(S) EXAB1   -  Exchange AB with scratch 1
4334           ** Name:(S) RCCD1   -  Recall CD into scratch 1
4335           ** Name:(S) STAB2   -  Store AB into scratch 2
4336           ** Name:(S) EXAB2   -  Exchange AB with scratch 2
4337           ** Name:(S) RCCD2   -  Recall CD into scratch 2
4338           ** Name:(S) STCD2   -  Store CD into scratch 2
4339           **
4340           ** Category:   MTHUTL
4341           **
4342           **
4343           ** Purpose:
4344           **      To use R0-R3 as scratch space for 15-form numbers.
```

```
4345                    **
4346                    ** Entry:
4347                    **      Either AB or CD has a 15-form to be transfered with
4348                    **      (R0,R1) or (R2,R3).
4349                    **
4350                    ** Exit:
4351                    **      Data transfer has taken place.
4352                    **
4353                    ** Calls:      (none)
4354                    **
4355                    ** Uses.......
4356                    **  Inclusive: nothing
4357                    **
4358                    ** Stk lvls:   0
4359                    **
4360                    ************************************************************************
4361                    ************************************************************************
4362 0D3D9 100   =STAB1  R0=A
4363 0D3DC AFC          ABEX   W
4364 0D3DF 101          R1=A
4365 0D3E2 AFC          ABEX   W
4366 0D3E5 01           RTN
4367 0D3E7 120   =EXAB1  AR0EX
4368 0D3EA AFC          ABEX   W
4369 0D3ED 121          AR1EX
4370 0D3F0 AFC          ABEX   W
4371 0D3F3 01           RTN
4372 0D3F5 119   =RCCD1  C=R1
4373 0D3F8 AF7          D=C    W
4374 0D3FB 118          C=R0
4375 0D3FE 01           RTN
4376 0D400 102   =STAB2  R2=A
4377 0D403 AFC          ABEX   W
4378 0D406 103          R3=A
4379 0D409 AFC          ABEX   W
4380 0D40C 01           RTN
4381 0D40E 122   =EXAB2  AR2EX
4382 0D411 AFC          ABEX   W
4383 0D414 123          AR3EX
4384 0D417 AFC          ABEX   W
4385 0D41A 01           RTN
4386 0D41C 11B   =RCCD2  C=R3
4387 0D41F AF7          D=C    W
4388 0D422 11A          C=R2
4389 0D425 01           RTN
4390 0D427 10A   =STCD2  R2=C
4391 0D42A AFF          CDEX   W
4392 0D42D 10B          R3=C
4393 0D430 AFF          CDEX   W
4394 0D433 01           RTN
4395 0D435              END
```

```
 +       Abs       0 #00000 -    92   238   283   751   811   840
 /       Abs       2 #00002 -    94   511   749   809
=1/X15   Abs   49982 #0C33E -   171
=1/X15S  Abs   49988 #0C344 -   174  3694
 A/B=0   Abs   50970 #0C71A -  1130
=AD15M   Abs   50022 #0C366 -   235  1366  1829
=AD15S   Abs   50025 #0C369 -   237   143
=AD2-12  Abs   50015 #0C35F -   231
=AD2-15  Abs   50019 #0C363 -   233
 ADD20   Abs   50083 #0C3A3 -   262   269
 ADD30   Abs   50132 #0C3D4 -   293   267
 ADD40   Abs   50142 #0C3DE -   297   271   274   294
 ADD50   Abs   50146 #0C3E2 -   299   303
 ADD60   Abs   50159 #0C3EF -   304   260   265   300
 ADD70   Abs   50172 #0C3FC -   309   307
 ADD80   Abs   50199 #0C417 -   322   305
 ADD90   Abs   50202 #0C41A -   323   311   315   317   320
=ADDF    Abs   50034 #0C372 -   243   284
=ADDONE  Abs   49968 #0C330 -   139  2962  3730
 AVMEME  Ext                -  2819
 BIASA-  Ext                -  2137
 BIASC+  Ext                -  2111
 BIASC-  Ext                -  2149
 BIG     Ext                -  2228
 BIG15   Abs   51062 #0C776 -  1215  1195  1210
 BIGSML  Abs   51043 #0C763 -  1209  1200
 BSERR   Ext                -  2654
 CHKmem  Ext                -  2698
 CKIX?   Abs   52106 #0CB8A -  2472  2466
 CKIXP   Abs   52122 #0CB9A -  2483  2474  2509
 CLREND  Abs   50956 #0C70C -  1100  1098
=CLRFRC  Abs   50932 #0C6F4 -  1091  1772  4068
 DBLSUB  Ext                -  3562
=DECP=C  Abs   51025 #0C751 -  1201
=DIV0    Abs   50374 #0C4C6 -   523
=DIV100  Abs   50426 #0C4FA -   552   522
=DIV120  Abs   50447 #0C50F -   560   557
 DIV130  Abs   50456 #0C518 -   563   565
 DIV140  Abs   50459 #0C51B -   564   562   569
=DIV15   Abs   50437 #0C505 -   556   554  2898
=DIVF    Abs   50360 #0C4B8 -   516   289
=DIVFCD  Abs   50363 #0C4BB -   519
 DIVOP   Abs   50749 #0C63D -   828   810
=DV15M   Abs   50348 #0C4AC -   509
=DV15S   Abs   50354 #0C4B2 -   511   182  3852
=DV2-12  Abs   50344 #0C4A8 -   506
=DV2-15  Abs   50348 #0C4AC -   508  1765
=DXP100  Abs   53119 #0CF7F -  3488
 DXP200  Abs   53184 #0CFC0 -  3539  3499
 DXP210  Abs   53219 #0CFE3 -  3561  3563
 DXP220  Abs   53221 #0CFE5 -  3562  3570
 DXP225  Abs   53256 #0D008 -  3576  3568
=DZ10    Abs   50398 #0C4DE -   535   533
 DZERO   Abs   50389 #0C4D5 -   531   524
=DZINF   Abs   50404 #0C4E4 -   538  3050
```

```
=DZP     Abs       3 #00003 -    82   548  2006  2580
=EFIELD  Abs       0 #00000 -    86   527   639   791   846   918  3034  3046
                                4093  4167  4230
 ENDINT  Abs   50997 #0C735 -  1141  1137
 ERRMSG  Abs   52085 #0CB75 -  2458  2452  2494
=EX-112  Abs   53060 #0CF44 -  3454
=EX-115  Abs   53064 #0CF48 -  3456
 EX15M   Ext                -  3837  3930
=EXAB1   Abs   54247 #0D3E7 -  4367  3911
=EXAB2   Abs   54286 #0D40E -  4381  4155  4162  4202
=EXP100  Abs   53113 #0CF79 -  3485  3467
 EXP110  Abs   53130 #0CF8A -  3494  3491
 EXP115  Abs   53136 #0CF90 -  3496  3508
=EXP12   Abs   53078 #0CF56 -  3462
 EXP120  Abs   53162 #0CFAA -  3517  3503
 EXP130  Abs   53164 #0CFAC -  3518  3522
 EXP131  Abs   53180 #0CFBC -  3524  3520
=EXP15   Abs   53082 #0CF5A -  3463
=EXP15M  Abs   53082 #0CF5A -  3464
=EXP16M  Abs   53110 #0CF76 -  3480  4135
 EXP240  Abs   53747 #0D1F3 -  3886  3881
 EXP400  Abs   53303 #0D037 -  3615  3523  3585
 EXP410  Abs   53279 #0D01F -  3604  3606
 EXP420  Abs   53282 #0D022 -  3605  3616
 EXP500  Abs   53310 #0D03E -  3631  3524  3610
 EXP520  Abs   53394 #0D092 -  3676  3660
 EXP521  Abs   53426 #0D0B2 -  3695  3693
 EXP530  Abs   53375 #0D07F -  3668  3664
 EXP540  Abs   53372 #0D07C -  3667  3669
 EXP550  Abs   53348 #0D064 -  3656  3647  3679
 EXP560  Abs   53355 #0D06B -  3659  3674
 EXP565  Abs   53331 #0D053 -  3642  3633
 EXP570  Abs   53333 #0D055 -  3643
 EXP600  Abs   53430 #0D0B6 -  3708  3657
 EXP700  Abs   53774 #0D20E -  3899  3897
 EXP710  Abs   53777 #0D211 -  3902  3476  3695  3964  4249
 EXP740  Abs   53447 #0D0C7 -  3718  3711
 EXP750  Abs   53456 #0D0D0 -  3723  3719
 EXP760  Abs   53469 #0D0DD -  3730  3725  3727
=EXP90   Abs   53088 #0CF60 -  3466  3460  4123
=FINITA  Abs   52483 #0CD03 -  2861  1126  1194  1668  4291
=FINITC  Abs   52495 #0CD0F -  2870  1675  4064
=FNPWDS  Abs   54208 #0D3C0 -  4289  2946  3028  3466
=FRAC15  Abs   50958 #0C70E -  1126
=FRC15F  Abs   50977 #0C721 -  1134  1127
 FRCITR  Abs   50983 #0C727 -  1136  1140
 GETSGN  Abs   52156 #0CBBC -  2506  2500
 GIVARG  Abs   51409 #0C8D1 -  1672  1691
=GMEME   Abs   52471 #0CCF7 -  2818  2752  2779
=HNDL20  Abs   52188 #0CBDC -  2556  2551
 HNDL30  Abs   52194 #0CBE2 -  2577  2554
 HNDL40  Abs   52211 #0CBF3 -  2583  2579
=HNDLFL  Abs   52169 #0CBC9 -  2547  2025
=HTRAP   Abs   52015 #0CB2F -  2415  2024
 HUGE20  Ext                -  2204
```

```
=IDIV12  Abs   51449 #0C8F9 -  1764
=IDIV15  Abs   51453 #0C8FD -  1765
 IDV30   Abs   51464 #0C908 -  1772  1767
=IF12A   Abs   51001 #0C739 -  1193
=INF*0   Abs   50695 #0C607 -   790   815
=INFR15  Abs   51005 #0C73D -  1194  1091  1134  4176
 INFS+   Abs   50821 #0C685 -   917   841
 INVNNj  Abs   52630 #0CD96 -  3036  4171
=INVNaN  Abs   50783 #0C65F -   894   529   641   793   921  3036
 IV50    Abs   50804 #0C674 -   904   907
 IVARG   Ext                -  1673
=IVP     Abs       4 #00004 -    83   910  2008  2015  2493  2578
 Ivn     Abs   54097 #0D351 -  4171  4104
=LGT12   Abs   53674 #0D1AA -  3828
=LGT15   Abs   53678 #0D1AE -  3829
 LGT50   Abs   53702 #0D1C6 -  3841  3832  3836
=LN1+12  Abs   52544 #0CD40 -  2943
=LN1+15  Abs   52548 #0CD44 -  2945
 LN1+X1  Abs   52586 #0CD6A -  2976  2955
 LN1+X2  Abs   52576 #0CD60 -  2962
=LN1+X4  Abs   52571 #0CD5B -  2954
 LN1+X5  Abs   52597 #0CD75 -  2984  2977
 LN1+X6  Abs   52724 #0CDF4 -  3111  2985
 LN1+X7  Abs   52751 #0CE0F -  3125  3116
 LN1+X8  Abs   52741 #0CE05 -  3120  3126
=LN1+XF  Abs   52561 #0CD51 -  2950  2947
 LN110   Abs   52759 #0CE17 -  3146  3059
=LN12    Abs   52605 #0CD7D -  3024
 LN140   Abs   52769 #0CE21 -  3163  3148
=LN15    Abs   52609 #0CD81 -  3026  3841  4084
 LN200   Abs   52706 #0CDE2 -  3092  3083  3097
 LN210   Abs   52720 #0CDF0 -  3101  3094
=LN30    Abs   52636 #0CD9C -  3039  2965  3029
 LN300   Abs   52780 #0CE2C -  3170  3128  3190
 LN310   Abs   52777 #0CE29 -  3169  3180
 LN320   Abs   52790 #0CE36 -  3174  3176
 LN330   Abs   52793 #0CE39 -  3175  3172
 LN400   Abs   52867 #0CE83 -  3250  3121  3243
 LN420   Abs   52864 #0CE80 -  3247  3234
 LN431   Abs   52899 #0CEA3 -  3277  3255
 LN450   Abs   52886 #0CE96 -  3264  3224
 LN460   Abs   52896 #0CEA0 -  3276  3285
 LN50    Abs   52668 #0CDBC -  3054  3044
 LN520   Abs   52943 #0CECF -  3311  3302  3315
 LN530   Abs   52964 #0CEE4 -  3329  3280  3313
 LN540   Abs   52958 #0CEDE -  3326  3330
 LN550   Abs   52971 #0CEEB -  3332  3327
 LN565   Abs   52989 #0CEFD -  3339  3089  3239  3337
=LNAP    Abs   53013 #0CF15 -  3379  3250  3636
 LNAP1   Abs   53030 #0CF26 -  3385  3389
 LNAP2   Abs   53055 #0CF3F -  3394  3387
=LNC10   Abs   53653 #0D195 -  3792  3300
 LNC10+  Abs   53650 #0D192 -  3791  3547  3849
 LNC20   Abs   53475 #0D0E3 -  3739  3277  3615
 LNC30   Abs   53542 #0D126 -  3766  3742
```

```
 LNC40    Abs   53565 #0D13D -  3770  3747
 LNC50    Abs   53586 #0D152 -  3774  3749
 LNC60    Abs   53605 #0D165 -  3778  3751
 LNC70    Abs   53622 #0D176 -  3782  3753
 LNC80    Abs   53637 #0D185 -  3786  3755
 LNC90    Abs   53533 #0D11D -  3762  3757
 LNINF    Abs   52619 #0CD8B -  3031  2948
=LNNEG    Abs   52624 #0CD90 -  3034  3055
 LNSUB    Abs   52507 #0CD1B -  2888  2984  3715
 LNSUB1   Abs   52515 #0CD23 -  2891  2895
 LNSUB2   Abs   52539 #0CD3B -  2899  2893
 LNSUB3   Abs   52533 #0CD35 -  2898  3102  3393
 LNZR     Abs   52650 #0CDAA -  3046
 LNZR1    Abs   52662 #0CDB6 -  3050  4247
 MAKE1    Ext                 -  3835  4055
 MATHR    Abs   50118 #0C3C6 -   282
=MESSG    Abs   52247 #0CC17 -  2647  2026
 MFWRNQ   Ext                 -  2775
=MOD12    Abs   51090 #0C792 -  1329
=MOD15    Abs   51094 #0C796 -  1331
 MOD50    Abs   51137 #0C7C1 -  1356  1687
 MODX<Y   Abs   51131 #0C7BB -  1349  1337
 MOved3   Ext                 -  2793
 MOveu3   Ext                 -  2745
=MP1-12   Abs   50230 #0C436 -   381
=MP15S    Abs   50240 #0C440 -   385
=MP2-12   Abs   50226 #0C432 -   379
=MP2-15   Abs   50234 #0C43A -   382  3931  4087
 MPY110   Abs   50266 #0C45A -   398   396
 MPY120   Abs   50271 #0C45F -   400   407
 MPY130   Abs   50280 #0C468 -   403   405
 MPY140   Abs   50283 #0C46B -   404   402
 MPY150   Abs   50300 #0C47C -   410   330  3339
 MSN15    Ext                 -  1663  4007
=MULTF    Abs   50246 #0C446 -   390   287
 NEGT     Abs   51219 #0C813 -  1446  1438
 NMNF     Abs   50924 #0C6EC -  1043  1038
 NOERR    Abs   52297 #0CC49 -  2701
 NOHALT   Abs   52262 #0CC26 -  2657  2649
 NONINT   Abs   54091 #0D34B -  4166
 NOTMOD   Abs       8 #00008 -    75  1281  1332  1407  1683
 NRM12    Abs   51643 #0C9BB -  2022  2002
=NRMLAB   Abs   50901 #0C6D5 -  1033  2088  3830  4013
=NRMLCD   Abs   51542 #0C956 -  1905  4021
=OKP      Abs       0 #00000 -    79  2093  2140  2418  2451  2465  2483  2490
                                2552  2657
 ONE15    Abs   54037 #0D315 -  4106  4097  4238
=OVF-15   Abs   53751 #0D1F7 -  3890  3572  3884  3940
=OVF15?   Abs   53729 #0D1E1 -  3870  3584  3950
=OVFL     Abs   51827 #0CA73 -  2186  2135
 OVFL20   Abs   51862 #0CA96 -  2208  2193
 OVFL30   Abs   51875 #0CAA3 -  2222  2211
 OVFL40   Abs   51880 #0CAA8 -  2226  2196  2216
 OVINF    Abs   51850 #0CA8A -  2201  2218  2223
=OVP      Abs       2 #00002 -    81  2191
```

```
 PICKUP  Abs   51037 #0C75D -  1206  1203
 PMUL    Abs   52998 #0CF06 -  3353  3278  3311
 PMUL1   Abs   52995 #0CF03 -  3352  3354
 PMUL2   Abs   53011 #0CF13 -  3358
 PWEEDS  Abs   50643 #0C5D3 -   725   239   386   512
 PWR40   Abs   53819 #0D23B -  3936  3934
 PWR60   Abs   53834 #0D24A -  3944  3946
 PWR80   Abs   53864 #0D268 -  3961  3958
 PWRTEN  Abs   53798 #0D226 -  3927  4073
 R=Y/2   Abs   51205 #0C805 -  1440  1435
 R>Y/2   Abs   51195 #0C7FB -  1436
 R?Y/2   Abs   51172 #0C7E4 -  1423  1462  1484
=RCCD1   Abs   54261 #0D3F5 -  4372
=RCCD2   Abs   54300 #0D41C -  4386  4063  4085  4229
 RD300   Abs   51381 #0C8B5 -  1625  1623
 RED0<Y  Abs   51367 #0C8A7 -  1618  1574
 RED50   Abs   51302 #0C866 -  1576
 RED90   Abs   51317 #0C875 -  1583  1579
=REDUCe  Abs   51282 #0C852 -  1567  1285  1336  1410
 REDX<Y  Abs   51370 #0C8AA -  1620  1588
=REM12   Abs   51151 #0C7CF -  1405
 REM120  Abs   51190 #0C7F6 -  1434  1427
=REM15   Abs   51155 #0C7D3 -  1406
 REM310  Abs   51241 #0C829 -  1465
 REM330  Abs   51251 #0C833 -  1480  1466
 REMP2   Abs   51402 #0C8CA -  1668  1664
 REMP20  Abs   51441 #0C8F1 -  1689  1676
 REMP5   Abs   51417 #0C8D9 -  1675  1669
 REMPWD  Abs   51383 #0C8B7 -  1660  1282  1334  1408
 REMX<Y  Abs   51225 #0C819 -  1455  1411
=RMD12   Abs   51066 #0C77A -  1279
=RMD15   Abs   51070 #0C77E -  1280
 RMOD30  Abs   51339 #0C88B -  1601  1610
 RMOD35  Abs   51342 #0C88E -  1602  1599
 RMOD40  Abs   51345 #0C891 -  1604  1606
=RND12+  Abs   51669 #0C9D5 -  2088
 RND120  Abs   51908 #0CAC4 -  2283  2278
 RND125  Abs   51927 #0CAD7 -  2297  2287  2344  2348
 RND130  Abs   51934 #0CADE -  2301  2292
 RND140  Abs   51944 #0CAE8 -  2311  2302  2335  2343  2349
 RND150  Abs   51950 #0CAEE -  2317  2280  2298  2334
 RND160  Abs   51969 #0CB01 -  2324  2321
 RND190  Abs   51976 #0CB08 -  2330  2304
 RND210  Abs   51990 #0CB16 -  2338  2285
 RND220  Abs   52003 #0CB23 -  2347  2339
 RND50   Abs   51692 #0C9EC -  2102  2089
=RNDNRM  Abs   51889 #0CAB1 -  2273  2106  2166
=RNDXCP  Abs   51678 #0C9DE -  2094  2004
 RSTDEF  Abs   52242 #0CC12 -  2595  2553  2813
 RSTEXP  Abs   51183 #0C7EF -  1430  1287  1341  1344  1346
 RSUBR   Abs   52431 #0CCCF -  2797  2801
 RTNDEF  Abs   52097 #0CB81 -  2468  2460  2476  2480  2491
 RTNX    Abs   51246 #0C82E -  1470
 SCRST0  Ext               -  2742  2789
 SETSB   Ext               -  3333  3728
```

```
 SETXM    Abs   50819 #0C683 -   911
 SGNOF*   Abs   50734 #0C62E -   818   750   756   830   843
 SGNS     Abs   51263 #0C83F -  1517  1333
=SHF10    Abs   50310 #0C486 -   443   411  1142  1431
 SHF20    Abs   50317 #0C48D -   446   450
 SHF30    Abs   50331 #0C49B -   451   445
 SHF40    Abs   50333 #0C49D -   453   447
 SHFD     Abs   51563 #0C96B -  1915  1909
 SHFD20   Abs   51577 #0C979 -  1920  1924
 SHFD30   Abs   51591 #0C987 -  1926  1919
 SHFD40   Abs   51593 #0C989 -  1927  1921
 SHFLAC   Ext                 -  3564
=SIGNAN   Abs   51478 #0C916 -  1817
 SNAPLC   Ext                 -  2711
 SPLA10   Abs   50895 #0C6CF -  1031  1029
 SPLC10   Abs   51536 #0C950 -  1902  1900
=SPLITA   Abs   50879 #0C6BF -  1025   380   606  1193  1864  2352
=SPLITC   Abs   51520 #0C940 -  1896   381
=SPLTAC   Abs   51508 #0C934 -  1862   232   507  1279  1330  1405  1764  1828
                             4003
 SQR-     Abs   50531 #0C563 -   638   624
=SQR12    Abs   50480 #0C530 -   606
=SQR15    Abs   50484 #0C534 -   608
=SQR15M   Abs   50484 #0C534 -   609
=SQR17    Abs   50515 #0C553 -   630   617
 SQR18    Abs   50541 #0C56D -   643   637
 SQR20    Abs   50563 #0C583 -   653   650
 SQR30    Abs   50587 #0C59B -   664   662
 SQR35    Abs   50601 #0C5A9 -   669   678
 SQR40    Abs   50607 #0C5AF -   672   674
 SQR50    Abs   50610 #0C5B2 -   673
=SQR70    Abs   50627 #0C5C3 -   714   633
 SQR75    Abs   50630 #0C5C6 -   715   570
 SQR80    Abs   50641 #0C5D1 -   719   716
 SQRXCP   Abs   50503 #0C547 -   619
=STAB1    Abs   54233 #0D3D9 -  4362  3459  3721  3908
=STAB2    Abs   54272 #0D400 -  4376
=STCD2    Abs   54311 #0D427 -  4390  4028
 SUBON1   Abs   49971 #0C333 -   140   137
=SUBONE   Abs   49959 #0C327 -   135  3909
 SVSUBR   Abs   52323 #0CC63 -  2719  2723
 SYSFLG   Ext                 -  2589
 Shf      Abs   50997 #0C735 -  1142  1041
 TRP30    Abs   52031 #0CB3F -  2425  2014  2419
 TRP=0    Abs   52072 #0CB68 -  2450  2442
 TRP=1    Abs   52128 #0CBA0 -  2489  2444
 TRP=2    Abs   52092 #0CB7C -  2464  2446
 TRPREG   Ext                 -  2431
 TSTRAP   Abs   52054 #0CB56 -  2439  2484
 TWON1    Abs   50871 #0C6B7 -   981   976   980
=TWONaN   Abs   50853 #0C6A5 -   971   927
 UNFL     Abs   51761 #0CA31 -  2148  2131
 UNFL10   Abs   51776 #0CA40 -  2155  2159
 UNFL20   Abs   51789 #0CA4D -  2163  2156
 UNFL40   Abs   51816 #0CA68 -  2176
```

=UNP	Abs	1	#00001	-	80	2169	2475	2499	
X&Y#S	Abs	50118	#0C3C6	-	280	742			
=X/Y15	Abs	49999	#0C34F	-	178				
X=XCP	Abs	50705	#0C611	-	800	736			
XCPARG	Abs	54224	#0D3D0	-	4296	4292			
XCPT30	Abs	51734	#0CA16	-	2129	2116			
XCPT50	Abs	51821	#0CA6D	-	2180	2125			
XCPT55	Abs	51756	#0CA2C	-	2142	2178	2181		
XFSET	Abs	52214	#0CBF6	-	2588	2581			
XLOG20	Abs	53713	#0D1D1	-	3849	3843			
XNNYCP	Abs	50834	#0C692	-	925	834			
=XYEX	Abs	50839	#0C697	-	961	825	982	4066	4071
Y&XXCP	Abs	50757	#0C645	-	832	803			
YCP;X#	Abs	50675	#0C5F3	-	744	741			
=YX050	Abs	53938	#0D2B2	-	4034	4030	4206	4211	
YX055	Abs	53947	#0D2BB	-	4053	4035			
YX056	Abs	53998	#0D2EE	-	4082	4056	4065	4072	
=YX060	Abs	54050	#0D322	-	4121	4090			
YX100	Abs	54059	#0D32B	-	4132	4122			
=YX2-12	Abs	53876	#0D274	-	4002				
=YX2-15	Abs	53882	#0D27A	-	4004				
YX200	Abs	54071	#0D337	-	4155	4032			
YX250	Abs	54082	#0D342	-	4162	4190			
YX30	Abs	53919	#0D29F	-	4020	4014			
YX400	Abs	54101	#0D355	-	4176	4158			
YX450	Abs	54123	#0D36B	-	4189	4179			
YX500	Abs	54128	#0D370	-	4195	4184			
YX550	Abs	54145	#0D381	-	4202	4183			
YX600	Abs	54166	#0D396	-	4211	4205			
YX700	Abs	54170	#0D39A	-	4229	4036	4164		
YX750	Abs	54191	#0D3AF	-	4240	4232			
YX800	Abs	54204	#0D3BC	-	4249	4241			
big	Abs	51883	#0CAAB	-	2228	2504			
e0^0	Ext			-	4235				
e0^NEG	Ext			-	4244				
e1^INF	Ext			-	4101				
eIF*ZR	Ext			-	792				
eIF-IF	Ext			-	919				
eIF/IF	Ext			-	847				
eINF^0	Ext			-	4095				
eINX	Ext			-	2456				
eLNO	Ext			-	3047				
eLOG-	Ext			-	3035				
eNEG^X	Ext			-	4169				
eOVFLW	Ext			-	2188				
eSQR-	Ext			-	640				
eUNFLW	Ext			-	2172				
eZRDIV	Ext			-	536				
eZR0/0	Ext			-	528				
exp710	Abs	53106	#0CF72	-	3476	3471			
ln1+X6	Abs	52601	#0CD79	-	2985	2981			
messg	Abs	51657	#0C9C9	-	2026	2017	2019		
mfERR	Abs	52256	#0CC20	-	2654	2699			
s=SGNS	Abs	9	#00009	-	74	1340	1356	1519	1523
sDZ	Abs	10	#0000A	-	71				

=sINFRD	Abs	10	#0000A	-	64	314	2192	2284			
=sINX	Abs	5	#00005	-	67						
sIV	Abs	11	#0000B	-	72						
=sIX	Abs	7	#00007	-	68	2094	2187	2276	2283	2421	2473	2496
					2550						
=sNEGRD	Abs	11	#0000B	-	65	233	316	2195	2210	2286	2338	
sOV	Abs	9	#00009	-	70						
sUN	Abs	8	#00008	-	69						
=sXCPT	Abs	4	#00004	-	66						
sY=INF	Abs	8	#00008	-	73	4010	4017	4096			
shf10	Abs	51186	#0C7F2	-	1431	1445	1447				
snapr*	Ext			-	2807						
splitA	Abs	52011	#0CB2B	-	2352	2943	3024	3454	3462	3828	
=uAD12	Abs	51490	#0C922	-	1827						
uMODES	Ext			-	1827	2077					
=uRES12	Abs	51604	#0C994	-	1993	1830					
=uRESNX	Abs	51645	#0C9BD	-	2023						
=uRESXT	Abs	51649	#0C9C1	-	2024	2007	2009				
=uRND12	Abs	51661	#0C9CD	-	2076						
=uRND>P	Abs	51663	#0C9CF	-	2077	2023					
x	Abs	1	#00001	-	93	286	385	842			
xyex	Abs	50745	#0C639	-	825	806	812				

Input Parameters

  Source file name is JT&MTH::MS

  Listing file name is JT/MTH:TI:ML::-1

  Object file name is JTXMTH:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
1            *     SSS   M   M  &     M   M  TTTTT  H   H
2            *    S   S  MM MM  & &    MM MM    T    H   H
3            *    S      M M M  & &    M M M    T    H   H
4            *     SSS   M M M   &     M M M    T    HHHHH
5            *        S  M   M  & & &  M   M    T    H   H
6            *    S   S  M   M  & & &  M   M    T    H   H
7            *     SSS   M   M  && &  M   M    T    H   H
8
9                    TITLE  Math Routines - Part 2 <831216.1618>
10 0D435             ABS    #0D435
11
12           ********************************************************************
13           ***                                                              ***
14           ***         MATH ROUTINES:     <SM&MTH>                          ***
15           ***                                                              ***
16           ***                                                              ***
17           ***  INPUTS:                                                     ***
18           ***                                                              ***
19           ***          -BINARY FUNCTIONS-                                  ***
20           ***                                                              ***
21           ***       2-12 digit numbers,then                                ***
22           ***             A HAS 12 DIGIT FORM                              ***
23           ***             C HAS 12 DIGIT FORM                              ***
24           ***                                                              ***
25           ***       2-15 digit numbers, then                               ***
26           ***             A: SIGN & EXP,  B: MANT                          ***
27           ***             C: SIGN & EXP,  D: MANT                          ***
28           ***                                                              ***
29           ***                                                              ***
30           ***                                                              ***
31           ***                                                              ***
32           ***           -UNARY FUNCTIONS-                                  ***
33           ***                                                              ***
34           ***      1-12 digit number, then                                 ***
35           ***             A: 12 DIGIT FORM                                 ***
36           ***                                                              ***
37           ***      1-15 digit number, then                                 ***
38           ***             A: SIGN & EXP                                    ***
39           ***             B: 15 digit MANTISSA                             ***
40           ***                                                              ***
41           ***                                                              ***
42           ***                                                              ***
43           ***                                                              ***
44           ***  OUTPUTS:   A: SIGN AND EXP                                  ***
45           ***             B: 15 digit MANTISSA                             ***
46           ***            SB: 0 if surely EXACT                             ***
47           ***            XM: 0 if no excpt'n (ZD, INV)                     ***
48           ********************************************************************
49           ***   STATUS BITS   Modified 10/15/82 to permit use of =uMODES
50           ***                 to set Deg/Rad Mode into S9.  The change:
51           ***                 roles of S6 and S9 swapped.
52           ***
53           =sRAD   EQU   9           RAD/DEG Mode in Trig
54            sXM    EQU   9           Local value for XM
55            sIX    EQU   7           Local Inexact flag
```

```
56              sINVRT EQU    8               Invert result
57              sTAN   EQU    6               Desire Tangent
58              sSGN   EQU    10              Sign of Result
59              sSGNT  EQU    11              Sign of TAN
60
61              sCOMP  EQU    8               Need complementary angle
62              sATAN  EQU    6               Desire ATAN
63              s+PI/2 EQU    11              Add PI/2
64             ****************************************************************
65
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) uTEST   -  Perform comparisons
                **
                ** Category:   MATH
                **
                ** Purpose:  User Real Comparisons -   <, >=, etc.
                **
                ** Entry:    P encodes predicate (see Predicate table).
                **           A:a  C:c  (Arg's are 12-dig forms a&c).
                **
                ** Exit:     Carry=Result (Set=TRUE), P=Cell# for pair, raises
                **           Invalid if Unordered and predicate contains one
                **           of ">" or "<" but not "?".   If invalid, subse-
                **           quent action based on user traps.
                **
                ** Calls:    TST12A, (Also uRESXT - if INVALID raised)
                **
                ** Alters (INC): A,B,C,D,P,XM,SB,sIX
                **
                ** Stk lvls: MAX(3,MESSG)
                **
                ** Algorithm:  See =TST15
                **
                **     Date     Programmer            Modification
                **   --------   ----------   ---------------------------------
                **   07/09/82   SB           Bugfix: HTRAP now works off of sIX
                **   02/07/83   SB           Update header.
                ****************************************************************
                ****************************************************************
 OD435 80F2 =uTEST  CPEX    2
 OD439 06           RSTK=C              Save Predicate
 OD43B 80F2         CPEX    2
 OD43F 7330         GOSUB   TST12A
 OD443 07           C=RSTK
 OD445 400          RTNC                Done if true.
 OD448 888          ?P#     8           Result  <,=, or > ?
 OD44B 92           GOYES   EXIT03      Yes, Test false. No exception.
 OD44D AA7          D=C     XS          Predicate:   D[XS]
 OD450 25           P=      5
 OD452 80F2         CPEX    2           [0101]   :   C[XS]
 OD456 0E27         C=C&D   XS
 OD45A 92A          ?C=0    XS          Predicate had "<" or ">"  ??
 OD45D 71           GOYES   EXIT03      No.
 OD45F D2    OOPS3  C=0     A
 OD461 20           P=      0
 OD463 3100         LC(2)   =eUNORC     Err Msg: Unordered compare
 OD467 D5           B=C     A
 OD469 20           P=      =IVP
 OD46B 847          ST=0    sIX         Always exact.
 OD46E 8E00         GOSUBL  =uRESXT     <JT&MTH>
         00
 OD474 03    EXIT03 RTNCC               CC=Result false
```

120

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) TST12A  -  Compare numbers: 12-Digit arg's A,C
                ** Name:(S) TST15   -  Compare numbers: 15-Digit arg's A/B, C/D
                **
                ** Category:   MTHUTL
                **
                ** Purpose:    Determine relationship between numbers a & c.
                **
                ** Entry:      TST12A:   12-digit arg's in A & C.
                **             TST15 :   15-digit arg's in A&B and C&D.
                **                P encodes predicate.
                **
                ** Exit:       Carry set=TRUE, P has the cell# associated with
                **             the number pair, arg's in 15-dig form unchanged.
                **
                ** Calls:      SPLTB,AFIN,CFIN,BIASA+,BIASC+,
                **             BIASA-, BIASC-
                **
                ** Alters (INC): P,A,B,C,D,CARRY
                ** Stk lvls:   1
                **
                ** NOTE:       Predicate (INPUT) & Cell# (OUTPUT) Table
                **
                **               Pred   9-bias   P       Cell   Cell#   P
                **              -----   ------  ---      ----   -----  ---
                **                <      0001    1        a<c    0001   1
                **                =      0010    2        a=c    0010   2
                **               <=      0011    3        a>c    0100   4
                **                >      0100    4        a?c    1000   8
                **               <>      0101    5
                **               >=      0110    6
                **                ?      1000    8       ["?" = Unordered]
                **               <?      1001    9
                **               =?      1010   10
                **               >?      1100   12
                **                #      1101   13
                **
                **               (Pred is  9-bias of the system token)
                **
                ** Algorithm:  Direct comparison of S,EXP, & MANTISSA.
                ** History:
                **
                **     Date      Programmer            Modification
                **    --------   ----------   ---------------------------------
                **    07/12/82   SB           Documented
                **    10/06/82   SB           Code Pack: Eliminate Proj Mode
                **    02/09/83   SB           Code Pack: Consolidate a=NaN tests
                **    02/25/83   SB           Code Pack: Eliminate GOTO LOGIC.
                ****************************************************************
                ****************************************************************
 0D476 7292 =TST12A GOSUB   SPLTB       12-digit entry
 0D47A AC7  =TST15  D=C     S
```

```
176 0D47D 80FF          CPEX    15          P-->D[S]
177 0D481 ACF           CDEX    S
178 0D484 75A0          GOSUB   BIASA+      Remove EXP bias
179 0D488 74B0          GOSUB   BIASC+
180 0D48C 28            P=      8           Cell# for unordered.
181 0D48E 7E62          GOSUB   AFIN
182 0D492 5A4           GONC    FINA        If a finite, go FINA
183
184 0D495 AFD           BCEX    W           AB: S,EXP     CD: MANT
185 0D498 96C           ?A#0    B
186 0D49B 81            GOYES   LOGIC       If a=NaN, go LOGIC
187 0D49D 924           ?A#B    XS
188 0D4A0 C0            GOYES   SIGNA       If c finite & a=INF, go SIGNA
189 0D4A2 96D           ?B#0    B
190 0D4A5 E0            GOYES   LOGIC       If c=NaN, go LOGIC
191 0D4A7 940           ?A=B    S
192 0D4AA 16            GOYES   AEQC        If same INF, go AEQC
193
194 0D4AC 948   SIGNA   ?A=0    S           Pigeonhole on sign of a
195 0D4AF 24            GOYES   AGTC
196 0D4B1 21    ALTC    P=      1           Cell# 1
197 0D4B3 80FF  LOGIC   CPEX    15          Cell# to C[S]
198 0D4B7 0E43          D=C&D   S           Truth value (0=false, else true)
199 0D4BB AFD           BCEX    W           Restore standard format
200 0D4BE 7B60          GOSUB   BIASA-      Restore exponent bias
201 0D4C2 7A70          GOSUB   BIASC-
202 0D4C6 ACD           BCEX    S
203 0D4C9 80DF          P=C     15          Cell# back to P
204 0D4CD AC2           C=0     S
205 0D4D0 ACD           BCEX    S
206 0D4D3 94F           ?D#0    S           Truth value to CARRY
207 0D4D6 20            GOYES   CTRICK
208 0D4D8 AC3   CTRICK  D=0     S
209 0D4DB 01            RTN
210
211 0D4DD 7000  FINA    GOSUB   =FINITC     c finite?
212 0D4E1 AFD           BCEX    W           A,B: S,EXP    C,D: MANT
213 0D4E4 521           GONC    FINAC
214 0D4E7 96D           ?B#0    B           c=NaN?
215 0D4EA 9C            GOYES   LOGIC
216 0D4EC 949   SIGNC   ?B=0    S           Pigeonhole on sign of c
217 0D4EF 2C            GOYES   ALTC
218 0D4F1 24    AGTC    P=      4           Cell# 4
219 0D4F3 6FBF          GOTO    LOGIC
220
221 0D4F7 2E    FINAC   P=      14
222 0D4F9 91A           ?C=0    WP          a=0 ?
223 0D4FC A0            GOYES   AISO
224 0D4FE 91F           ?D#0    WP          a#0.    c#0 ?
225 0D501 01            GOYES   ACNOTO
226 0D503 58A           GONC    SIGNA       BET. a#0 and c=0.
227 0D506 91F   AISO    ?D#0    WP          a=0.    c#0 ?
228 0D509 3E            GOYES   SIGNC
229 0D50B 22    AEQC    P=      2           Cell# 2
230 0D50D 65AF          GOTO    LOGIC
```

```
231 0D511 944   ACNOTO ?A#B    S          Different signs?
232 0D514 89           GOYES   SIGNA      Yes. Pigeonhole on sign a
233 0D516 8B0          ?A>B    A          EXP(a) > EXP(c) ?
234 0D519 39           GOYES   SIGNA      Sign check
235 0D51B 8B4          ?A<B    A          EXP(a) < EXP(c) ?
236 0D51E EC           GOYES   SIGNC      Sign check
237 0D520 997          ?C>D    WP         MANT(a) > MANT(c) ?
238 0D523 98           GOYES   SIGNA      Sign check
239 0D525 993          ?C<D    WP         MANT(a) < MANT(c) ?
240 0D528 4C           GOYES   SIGNC      Sign check
241 0D52A 50E          GONC    AEQC       BET
242
243
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) BIASA+  -  Add Exp bias to A
                ** Name:    BIASA-  -  Remove Exp bias from A
                ** Name:(S) BIASC+  -  Add Exp bias to C
                ** Name:    BIASC-  -  Remove Exp bias from C
                **
                ** Category:   MTHUTL
                **
                ** Purpose:    Add (or remove) EXP bias [50000] to 15-dig Num
                **
                ** Entry:      15-digit number in A&B or C&D, DEC Mode.
                **
                ** Exit:       Unbiased or biased exponent, P=4.
                **
                ** Uses (INC): P, and A[A] (or C[A])
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **     Date     Programmer            Modification
                **   --------   ----------   ---------------------------------
                **   02/10/83   SB           Removed =BIASAC entry.
                **
                **
                ** NOTE:       BIASA+ = BIASA-      (EXP+50000+50000=EXP)
                ****************************************************************
                ****************************************************************
0D52D           =BIASA-
0D52D 24        =BIASA+ P=      4
0D52F B04               A=A+1   P
0D532 B04               A=A+1   P
0D535 B04               A=A+1   P
0D538 B04               A=A+1   P
0D53B B04               A=A+1   P
0D53E 01                RTN
0D540           =BIASC-
0D540 24        =BIASC+ P=      4
0D542 B06               C=C+1   P
0D545 B06               C=C+1   P
0D548 B06               C=C+1   P
0D54B B06               C=C+1   P
0D54E B06               C=C+1   P
0D551 01                RTN
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) MSN12   -  Find most significant NaN, 12-Dig arg's
                ** Name:(S) MSN15   -  Find most significant NaN, 15-Dig arg's
                **
                ** Category:   MTHUTL
                **
                ** Purpose:    For 2-arg functions return most significant NaN.
                **
                ** Entry:      [A,B]: x     [C,D]: y    (15-digit forms)
                **
                ** Exit:       CC - Neither x nor y is Nan, reg's not altered.
                **             CS - [A,B] has most significant NaN.
                **
                ** Calls:      SPLTB,AFIN,CFIN,=TWONaN,SWAPXY.
                **
                ** Alters:     Carry.  If exit CS, also registers A,B,C,D.
                **
                ** Stack lvls: 1
                **
                ** History:
                **
                **      Date      Programmer            Modification
                **    --------    ----------    ---------------------------------
                **     9/23/82    SB            Name change and 12-digit entry
                **    10/04/82    SB            Code pack - Change near IX
                ****************************************************************
                ****************************************************************
0D553 75B1  =MSN12  GOSUB   SPLTB
0D557 75A1  =MSN15  GOSUB   AFIN
0D55B 502           GONC    FX          If x finite, go FX
0D55E 926           ?A#C    XS
0D561 41            GOYES   FY          Else if y finite, go FY
0D563 968           ?A=0    B
0D566 F1            GOYES   IX          If x=INF, go IX
0D568 96A           ?C=0    B
0D56B 00            RTNYES              Else if y=INF, return x=NaN
0D56D 8E00          GOSUBL  =TWONaN     Both NaN's, get most significant
      00
0D573 02            RTNSC
0D575 96C   FY      ?A#0    B
0D578 00            RTNYES              If (x,y)=(NaN,Fin), return x
0D57A 03    CCEXIT  RTNCC               No NaN
0D57C 8E00  FX      GOSUBL  =FINITC
      00
0D582 500           RTNNC               No NaN
0D585 96A   IX      ?C=0    B
0D588 2F            GOYES   CCEXIT      If y=INF, No NaN's, go CCEXIT.
0D58A 7D74          GOSUB   SWAPXY      Return y(=NaN)
0D58E 02            RTNSC
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) CLASSA  -  Classification of numeric arg
                **
                ** Category:   MTHUTL
                **
                ** Purpose:    User classification of numeric argument
                **
                ** Entry:      12-digit argument x in A.
                **
                ** Exit:       12-digit y=CLASS(x) in C;   -6<=y<=6
                **
                ** Calls:      AFIN,MAKE1
                **
                ** Alters (INC): A,C,P,CARRY
                **
                ** Stk lvls:   1
                **
                ** Detail:     Sign(y) = Sign(x)
                **             Mag(y)  = 1,2,3,4,5,6 (below)
                **
                **          ------------------------------------
                **         |              x                |MAG(y)|
                **         |===============================|======|
                **         |             zero              |  1   |
                **         |-------------------------------|------|
                **         |          Denormalized         |  2   |
                **         |-------------------------------|------|
                **         |           Normalized          |  3   |
                **         |-------------------------------|------|
                **         |           Infinity            |  4   |
                **         |-------------------------------|------|
                **         |           Quiet NaN           |  5   |
                **         |-------------------------------|------|
                **         |        Signalling NaN         |  6   |
                **          ------------------------------------
                **
                **     DATE     Programmer          Modification
                **    --------  ----------  ---------------------------------
                **     6/01/82   SB          Documented
                **    10/25/82   SB          Code Pack:  Use MAKE1
                **    01/06/83   SB          SR# 30 - Distinguish Sig NaN.
                **    02/07/83   SB          Update header.
                ****************************************************************
                ****************************************************************
 OD590 7A35 =CLASSA GOSUB   MAKE1       Init result (c=+1, or -1)
 OD594 AC6          C=A     S
 OD597 7561         GOSUB    AFIN       x finite?
 OD59B 421          GOC     NONFIN
 OD59E 958          ?A=0    M          Zero?
 OD5A1 00           RTNYES                                  (C=1)
 OD5A3 B06          C=C+1   P
 OD5A6 908          ?A=0    P          Denormalized?
 OD5A9 00           RTNYES                                  (C=2)
```

```
  399 0D5AB 551          GONC    C+1RTN      BET; Normal non-zero    (C=3)
  400 0D5AE 304  NONFIN  LCHEX   4
  401 0D5B1 968          ?A=0    B           x=INF?
  402 0D5B4 00           RTNYES                                      (C=4)
  403 0D5B6 B06          C=C+1   P
  404 0D5B9 A6C          A=A-1   B
  405 0D5BC 968          ?A=0    B           Quiet NaN?
  406 0D5BF 00           RTNYES                                      (C=5)
  407 0D5C1 B06  C+1RTN  C=C+1   P           Signalling NaN
  408 0D5C4 01           RTN                                         (C=6)
  409
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) EX12    -  Return exponent of 12-dig arg
                ** Name:(S) EX15M   -  Return exponent of 15-dig arg (XM=SB=0)
                ** Name:(S) EX15S   -  Return exponent of 15-dig arg
                ** Name:(S) EXF     -  Return exponent of finite 15-dig arg
                **
                ** Category:   MATH
                **
                ** Purpose:    Returns the exponent of given argument.
                **
                ** Entry:
                **              EX12:  12-digit arg in A.
                **              EX15M: 15-digit arg in A&B.
                **              EX15S: 15-digit arg in A&B.
                **              EXF:   15-dig finite arg in A&B.
                **
                ** Exit:       A&B: y=EXPONENT(x) 15-digit form
                **
                ** Calls:      SPLTA,XMOSBO,AFIN,=DZ10
                **
                ** Alters (INC): A,B,SB,XM,P,CARRY
                **
                ** Stk lvls:   1
                **
                ** History:
                **
                **     Date     Programmer            Modification
                **   --------   ----------   --------------------------------
                **    6/01/82   SB           Documented.
                **    9/22/82   SB           EXPONENT(0) raises DVZ
                **   10/08/82   SB           Code Pack: Tighter loop
                **   12/09/82   SB           Improve Comments
                **   02/07/83   SB           Update Header.
                **   03/31/83   SB           Dedicated err msg: "EXPONENT(0)"
                ********************************************************************
                ********************************************************************
OD5C6 7C31      =EX12   GOSUB   SPLTA
OD5CA 7350      =EX15M  GOSUB   XMOSBO
OD5CE 7E21      =EX15S  GOSUB   AFIN
OD5D2 5C0               GONC    EXF             If x=Finite, go EXF
OD5D5 96C               ?A#0    B
OD5D8 12                GOYES   EXIT02          If x=NaN, Return it.
OD5DA AC0               A=0     S
OD5DD 03                RTNCC                   If |x|=Inf, Return +INF

OD5DF           =EXF                                    **FINITE ARGUMENT**
OD5DF AC1               B=0     S               Ensure zero field.
OD5E2 97D               ?B#0    W
OD5E5 61                GOYES   XNONO          If x not zero, go XNONO
OD5E7 AC0               A=0     S
OD5EA BCC               A=-A-1  S
```

```
  465 0D5ED 20          P=     0
  466 0D5EF 3100        LC(2)  =eEXPO          MSG:  "EXPONENT(0)"
  467 0D5F3 8E00        GOSUBL =DZINF          Answer = -INF,  raise DVZ
            00
  468 0D5F9 03    EXIT02 RTNCC
  469
  470 0D5FB AF1   XNONO  B=0    W
  471 0D5FE D8           B=A    A
  472 0D600 C4           A=A+A  A              Neg Exponent?
  473 0D602 AF0          A=0    W
  474 0D605 570          GONC   EXLOOP         If Not negative, Go EXLOOP.
  475 0D608 BCC          A=-A-1 S
  476 0D60B F9           B=-B   A
  477
  478 0D60D 815   EXLOOP BSRC   W              Rotate digit to MANT
  479 0D610 E4           A=A+1  A              Bump exponent
  480 0D612 8AD          ?B#0   A              Done?
  481 0D615 8F           GOYES  EXLOOP         No.
  482
  483 0D617 BF5          BSR    W              Shift MANT past sign field
  484 0D61A CC           A=A-1  A              Adjust exponent
  485 0D61C 822   SBOCC  SB=0                  Exact result.
  486 0D61F 03           RTNCC
  487
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                ** Name:(S) SQRSAV  -  SQR for Chain calculations.
                ** Name:(S) ORXM    -  Set XM if sXM=1 and Set SB if sIX=1
                ** Name:(S) ORSB    -  Set SB if sIX=1
                ** Name:(S) SETSB   -  Set SB
                **
                ** Category: MATH UTILITY
                **
                ** Purpose:  SQRSAV-Puts XM & SB into status bits sXM & sIX,
                **                  calls SQR15M, and falls into ORXM which
                **                  establishes XM<--XM OR sXM, SB<--SB OR sIX.
                **                  This preserves exactness in SB and
                **                  exceptions in XM thru a call to SQR15M.
                **
                ** Entry:    SQRSQV:15-Digit arg in A and B.
                **                  DEC Mode
                **                  XM Set if previous exception.
                **                  SB Set if previous inexact calculation
                **
                ** Exit:     SQR(Arg) in 15-digit form in A and B
                **           DEC Mode
                **           XM Set if previous exception or SQR exception.
                **           SB Set if previous inexact or SQR inexact.
                **
                ** Alters:   A,B,C,P,SB,XM,CARRY, and status bits sIX,sXM
                **
                ** Calls:    SAVEXM,SQR15M,SETXM
                **
                ** Stack Levels: 1
                **
                ** History:
                **
                **     Date     Programmer            Modification
                **   --------   ----------   ---------------------------------
                **   11/02/83   SB           Documented
                ****************************************************************
                ****************************************************************
0D621 821    XMOSBO XM=0               No invalid or zero divide
0D624 822           SB=0
0D627 01            RTN


0D629 7630  =SQRSAV GOSUB  SAVEXM        SB-->sIX & XM-->sXM
0D62D 8E00          GOSUBL =SQR15M       Square Root: [A,B]
      00
0D633 869   =ORXM   ?ST=0  sXM           Set XM if sXM=1
0D636 60            GOYES  ORSB
0D638 7010          GOSUB  SETXM
0D63C 867   =ORSB   ?ST=0  sIX           Set SB if sIX=1
0D63F 00            RTNYES
0D641 AC2   =SETSB  C=0    S
0D644 B46           C=C+1  S
0D647 BC6           CSR    S
```

```
  542 0D64A 01           RTN
  543
  544 0D64C 00    SETXM  RTNSXM
  545
```

```
546                     EJECT
547                 ****************************************************************
548                 ****************************************************************
549                 ** Name:(S) SAVGSB  -  Put SB into sINX
550                 ** Name:(S) ORGSB   -  Set SB if sINX=1
551                 ** Name:(S) SAVEXM  -  Put XM into sXM & SB into sIX
552                 ** Name:(S) SAVESB  -  Put SB into sIX
553                 **
554                 ** Category: MATH UTILITY
555                 **
556                 ** Purpose:  Routines save and restore SB and XM from status
557                 **
558                 ** Entry:    See description above
559                 **
560                 ** Exit:     See description above
561                 **
562                 ** Alters:   SAVGSB - CARRY, status sINX
563                 **           ORGSB  - C[S], SB, CARRY
564                 **           SAVEXM - CARRY, status sIX,sXM
565                 **           SAVESB - CARRY, status sIX
566                 **
567                 ** Calls:    Nothing
568                 **
569                 ** Stack Levels: 0
570                 **
571                 ** History:
572                 **
573                 **     Date     Programmer            Modification
574                 **   --------   ----------   ---------------------------------
575                 **   11/02/83   SB           Documented
576                 ****************************************************************
577                 ****************************************************************
578 0D64E 840    =SAVGSB ST=0   =sINX          SB-->sINX
579 0D651 832            ?SB=0
580 0D654 00             RTNYES
581 0D656 850            ST=1   =sINX
582 0D659 01             RTN
583                 ****************************************************************
584 0D65B 860    =ORGSB  ?ST=0  =sINX          Set SB if sINX=1
585 0D65E 00             RTNYES
586 0D660 50E            GONC   SETSB          BET.
587                 ****************************************************************
588 0D663 849    =SAVEXM ST=0   sXM            XM-->sXM
589 0D666 831            ?XM=0
590 0D669 50             GOYES  SAVESB
591 0D66B 859            ST=1   sXM
592 0D66E 847    =SAVESB ST=0   sIX            SB-->sIX
593 0D671 832            ?SB=0
594 0D674 00             RTNYES
595 0D676 857            ST=1   sIX
596 0D679 01             RTN
597
```

```
598                EJECT
599             ****************************************************************
600             ****************************************************************
601             **
602             ** Name:(S) ARG12   -  Return Arg of X+iY (12-dig args)
603             ** Name:(S) ARG15   -  Return Arg of X+iY (15-dig args)
604             ** Name:(S) ARGF    -  Return Arg of X+iY (15-dig finite args)
605             **
606             ** Category:   MATH
607             **
608             ** Purpose:    Argument of X+iY.  Used by ANGLE.
609             **
610             ** Entry:      ARG12:  12-Dig args-  A:X,  C:Y, =sRAD
611             **             ARG15:  15-Dig args- AB:X, CD:Y, =sRAD
612             **             ARGF :  15-Dig finite args- AB:Y, CD:X, =sRAD
613             **
614             ** Exit:       A&B: ARG(X,Y)
615             **
616             ** Calls:      SPLTB,MSN15,AFIN,SWAPXY,=DV2-15,SAVGSB,
617             **             ATAN15,ORGSB,PI/2D,=ADDF,XMOSBO.
618             **
619             ** Alters (INC): A,B,C,D,R0,R1,P,sIX,=sINX,sCOMP,sATAN,sSGN,
620             **               =sRAD,s+PI/2,SB,XM
621             **
622             ** Stk lvls:   2
623             **
624             ** Algorithm:  Weed special cases, call  ATAN15(Y/X)
625             **
626             ** History:
627             **
628             **     Date     Programmer            Modification
629             **   --------   ----------   ---------------------------------
630             **    6/30/82   SB           sAFFIN used in place of P
631             **   10/06/82   SB           Code Pack: Eliminate Proj Mode,
632             **                           Also bugfix (X,Y)=(FINITE,INF).
633             **   11/15/82   SB           Bugfix: ANGLE(0,0)=0 is EXACT.
634             **   02/07/83   SB           Update header.
635             **   11/02/83   SB           Additional Documentation
636             ****************************************************************
637             ****************************************************************
638 0D67B 7D80 =ARG12  GOSUB  SPLTB
639 0D67F 7E9F =ARG15  GOSUB  XMOSBO         XM=0, SB=0
640 0D683 70DE         GOSUB  MSN15
641 0D687 400          RTNC                  If any NaN, return a NaN.
642 0D68A 100          R0=A                  Save sign(X)
643 0D68D 109          R1=C                  Save sign(Y)
644 0D690 7C60         GOSUB  AFIN
645 0D694 7373         GOSUB  SWAPXY         Y:AB   X:CD
646 0D698 571          GONC   XFINIT         If X finite, go XFINIT.
647
648 0D69B 926          ?A#C   XS             (INF,?).
649 0D69E 91           GOYES  DIVYX          If Y finite, go DIVYX.
650 0D6A0 68A0         GOTO   IVARG          (INF,INF).  Error: Inv Arg.
651
652 0D6A4 97F  =ARGF   ?D#0   W
```

```
653 0D6A7 01          GOYES  DIVYX        If X#0, go DIVYX.
654 0D6A9 97D         ?B#0   W
655 0D6AC B0          GOYES  DIVYX        If Y#0, go DIVYX.
656 0D6AE 01          RTN                 (0,0).  Return Y.
657
658 0D6B0 7C40 XFINIT GOSUB  AFIN         (Finite, ?)
659 0D6B4 5FE         GONC   ARGF         If Y finite, go ARGF.
660 0D6B7 8E00 DIVYX  GOSUBL =DV2-15
          00
661 0D6BD 7D8F        GOSUB  SAVGSB       Save divide Exactness (=sINX)
662 0D6C1 79F4        GOSUB  ATAN15       ATAN(Y/X).
663 0D6C5 729F        GOSUB  ORGSB        Set SB if inexact divide.
664 0D6C9 118         C=R0                Fetch sign(X)
665 0D6CC 94A         ?C=0   S
666 0D6CF 00          RTNYES              If Sign(X)="+",  ATAN(Y/X).
667
668 0D6D1 119         C=R1                Fetch sign(Y)
669 0D6D4 2E          P=     14
670 0D6D6 A92         C=0    WP
671 0D6D9 AF7         D=C    W            D[S]=Sign(Y)
672 0D6DC 879         ?ST=1  =sRAD
673 0D6DF 21          GOYES  MAKEPI       If RAD Mode, go MAKEPI
674 0D6E1 E7          D=D+1  A            Create 180 (+/-)
675 0D6E3 E7          D=D+1  A            Exp=2
676 0D6E5 2D          P=     13
677 0D6E7 3181        LCHEX  18
678 0D6EB AFF         CDEX   W
679 0D6EE 5B0         GONC   ARGEX        BET.     CD = SGN(Y)*180
680 0D6F1 7584 MAKEPI GOSUB  PI/2D
681 0D6F5 A77         D=D+D  W            SGN(Y)*3.14159265358980
682 0D6F8 CF          D=D-1  A            SGN(Y)*3.14159265358979 (BETTER)
683 0D6FA 8C00 ARGEX  GOLONG =ADDF        NOTE: Never 0, ==> Round mode
          00
684                                             not important.
685
```

```
686                     EJECT
687               *=============================================================
688               *====================  JUMP TABLE  ===========================
689 0D700 8C00    AFIN    GOLONG =FINITA          JT%MTH    RTNCC IF X FINITE
          00
690 0D706 8C00    SPLTA   GOLONG =SPLITA          JT%MTH
          00
691 0D70C 8C00    SPLTB   GOLONG =SPLTAC          JT%MTH    SPLIT BOTH X & Y
          00
692               *SWAPXY GOLONG =XYEX  (Moved-BS)JT%MTH    SWAP X & Y
693 0D712 6000    =STAB1X GOTO   =STAB1           JT%MTH    STORE X IN R0,R1
694               *RCCD1X GOTO   =RCCD1 (Moved-BS)JT%MTH    FETCH Y FROM R0,R1
695               *=============================================================
696               *=============================================================
697
```

```
698                EJECT
699             ********************************************************************
700             ********************************************************************
701             **
702             ** Name:(S) SIN12   -  Trig: Sine of 12-dig arg
703             ** Name:(S) COS12   -  Trig: Cosine of 12-dig arg
704             ** Name:(S) TAN12   -  Trig: Tangent of 12-dig arg
705             ** Name:(S) SIN15   -  Trig: Sine of 15-dig arg
706             ** Name:(S) COS15   -  Trig: Cosine of 15-dig arg
707             ** Name:(S) TAN15   -  Trig: Tangent of 15-dig arg
708             **
709             ** Category:   MATH
710             **
711             ** Purpose:    SINE, COSINE, & TANGENT
712             **
713             ** Entry:      SIN12,COS12,TAN12 - Standard Math, 12-dig arg'ts
714             **             SIN15,COS15,TAN15 - Standard Math, 15-dig arg'ts
715             **
716             **                  All entries assume Status bit =sRAD encodes
717             **                  the desired angle mode (SET=RAD MODE)
718             **
719             ** Exit:       A&B: 15-digit result.  COS & SIN entries also
720             **                  produce TAN (or COT) magnitude in R0&R1.
721             **
722             ** Calls:      SPLTA,AFIN,SHFRAC,SHFRBD,PI/4,TWO*,DBLSUB,
723             **             SHFLAC,FLIP8,FLIP10,FLIP11,GETCON,=MULTF,=1/X15,
724             **             =DIVFCD,STAB1X,RCCD1X,=MP2-15,=ADDONE,=SQR15,
725             **             FUDGE.
726             **
727             ** Alters (INC) : A,B,C,D,R0,R1,P,SB,XM,CARRY, and
728             **             Status bits -  sIX, sINVRT, sTAN, sSGN, sSGNT.
729             **             Current Value:  7     8      6    10     11
730             **
731             ** Stk lvls:   2
732             **
733             ** Algorithm:
734             **     The absolute value of the argument is dbl word reduced
735             **     by 2*Pi (or 360), then by Pi/2, and Pi/4 to obtain
736             **     0<=Phi<=Pi/4.  A pseudo divide produces (X,Y) with
737             **     0<=Y<=X and TAN(Phi) = Y/X .  Formulas;
738             **            TAN(Phi) = Y/X
739             **            SIN(Phi) = 1/SQRT(1+(X/Y)^2)
740             **            COS(Phi) = 1/SQRT(1+(Y/X)^2)
741             **     Related back to the argument via STATUS bits.
742             **
743             ** sIX    (7) : Local exactness.  Not set=Exact. (INEXACT flag)
744             ** sINVRT (8) : If set, use X/Y instead of Y/X   (INVERT flag)
745             ** sTAN   (6) : If not set, TAN is desired       (TAN flag)
746             ** sSGN   (10): Sign of result                  (SGN  flag)
747             ** sSGNT  (11): Sign of TAN                     (SGNT flag)
748             **
749             ** History:
750             **
751             **     Date     Programmer            Modification
752             **   --------   ----------   ---------------------------------
```

```
753                ;**    7/15/82   SB          Fix to sign of 0 for COS(90), etc.
754                ;**    8/12/82   SB          Bugfix: Neg Exp in Radian Mode.
755                ;**   10/29/82   SB          Pack: INIT rearrangement
756                ;**   12/09/82   SB          Improve Comments, Label Change
757                ;**                         TRG150->REDUCE, Code Pack
758                ;**   12/14/82   SB          Label changes, code Pack in area
759                ;**                         where exactness established.
760                ;**   02/10/83   SB          Code Pack: Put XTENDE in line.
761                ;**   03/31/83   SB          Error msg change: TAN=INF replaces
762                ;**                         previous TAN or SEC=INF.
763                ;*****************************************************************
764                ;*****************************************************************
765 0D716 7CEF =SIN12  GOSUB  SPLTA
766 0D71A 858  =SIN15  ST=1   sINVRT         SIN:  (INVERT=YES)
767 0D71D 6A00        GOTO   TRG10                 (TAN=NO)
768
769 0D721 71EF =COS12  GOSUB  SPLTA
770 0D725 848  =COS15  ST=0   sINVRT         COS:  (INVERT=NO)
771 0D728 856   TRG10  ST=1   sTAN                 (TAN=NO)
772 0D72B 6D00        GOTO   TRG50
773
774 0D72F 73DF =TAN12  GOSUB  SPLTA
775 0D733 848  =TAN15  ST=0   sINVRT         TAN:  (INVERT=NO)
776 0D736 846         ST=0   sTAN                  (TAN=YES)
777
778 0D739 74EE  TRG50  GOSUB  XMOSBO         INIT: XM=0, SB=0
779 0D73D 7FBF        GOSUB  AFIN
780 0D741 531         GONC   TRGF                   If finite, go TRGF
781 0D744 96C         ?A#0   B
782 0D747 00          RTNYES                        If NaN, send it back.
783 0D749       OOPS4                               If INF, error
784 0D749 20   =IVARG  P=     0
785 0D74B 3100        LC(2)  =eIVARG               "Invalid Argument"
786 0D74F 8C00  BLDNAN GOLONG =INVNaN
          00
787
788 0D755 AF2   TRGF   C=0    W                       [FINITE ARG]
789 0D758 108         R0=C                  Low order MANT = zero
790 0D75B 84A         ST=0   sSGN           (SGN=+)
791 0D75E 84B         ST=0   sSGNT          (SGNT=+)
792 0D761 878         ?ST=1  sINVRT         SINE?
793 0D764 50          GOYES  TRG100
794 0D766 B46         C=C+1  S              flag: Retain COS entry.  Used
795 0D769 109   TRG100 R1=C                    for proper sign of COS(90)
796
797 0D76C AC1         B=0    S              Ensure zero field
798 0D76F 847         ST=0   sIX
799 0D772 979         ?B=0   W
800 0D775 50          GOYES  TRG125         If arg zero, EXACT result.
801 0D777 857         ST=1   sIX
802 0D77A 948   TRG125 ?A=0   S              Argument sign="+" ?
803 0D77D D0          GOYES  TRG130         Yep, go TRG130
804 0D77F 85B         ST=1   sSGNT          TAN(a)=-TAN(-a)
805 0D782 868         ?ST=0  sINVRT
806 0D785 50          GOYES  TRG130         COS(a)=COS(-a)
```

```
807 0D787 85A          ST=1   sSGN          SIN(a)=-SIN(-a)
808 0D78A      TRG130
809 0D78A AC0          A=0    S             |Arg|
810 0D78D 25           P=     5             Extend EXP sign [6-dig EXP]
811 0D78F A80          A=0    P
812 0D792 D6           C=A    A
813 0D794 C6           C=C+C  A
814 0D796 550          GONC   TRG135
815 0D799 A0C          A=A-1  P
816 0D79C AF6   TRG135 C=A    W             Exp--->C
817 0D79F AF4          A=B    W             High Mant--->A
818
819
820
821
822            ****************************************************************
823            *      PREPARE FOR FIXED POINT 31-DIGIT TRIG REDUCTION         *
824            *   Make argument alignment agree with reduction constant      *
825            *               [0360000...] or [062831...]                    *
826 0D7A2 879          ?ST=1  =sRAD         If RAD MODE: M=[0x.xxx...],
827 0D7A5 D1           GOYES  REDUCE          Exp=E, &  go REDUCE.
828
829 0D7A7 A1E          C=C-1  WP
830 0D7AA 90E          ?C#0   P             If X<10:  M=[0x.xxx...],
831 0D7AD D7           GOYES  TRG181          Exp=E-1 & skip Reduce.
832
833 0D7AF A1E          C=C-1  WP            If X>=100:  M=[0xxx.xxx...],
834 0D7B2 5F0          GONC   REDUCE          Exp=E-2, & go REDUCE.
835
836 0D7B5 B16          C=C+1  WP
837 0D7B8 128          CROEX                ELSE 10=<X<100: M=[00xx.xxx...],
838 0D7BB 7293         GOSUB  SHFRAC          Exp=E-1, & begin REDUCE.
839 0D7BF 128          CROEX
840            ****************************************************************
841
842
843
844            ****************************************************************
845            *       Generate reduction constant if arg not small           *
846 0D7C2 90E   REDUCE ?C#0   P             Neg EXP?
847 0D7C5 56           GOYES  TRG181         Yes. Skip 2*pi & pi/2 reduce
848 0D7C7 7133         GOSUB  DBLPI4         Fetch dbl pi/4 or 45
849 0D7CB 7963         GOSUB  TWO*           Doubling:  2*pi or 360  (B,D)
850 0D7CF 7563         GOSUB  TWO*
851 0D7D3 7163         GOSUB  TWO*
852 0D7D7 7483         GOSUB  SHFRBD         Shift 360 (or 2*pi) right
853            ****************************************************************
854
855
856
857            ****************************************************************
858            *                  360 or 2*PI Reduction                       *
859 0D7DB 128   RED360 CROEX                 Mant--->(A,C)  Exp--->R0
860 0D7DE 7BF2  TRG155 GOSUB  DBLSUB         Double subtract
861 0D7E2 5BF          GONC   TRG155         Repeat until negative
```

```
 862 0D7E5 7D53          GOSUB  SHFLAC          Align for next pass
 863 0D7E9 128           CROEX
 864 0D7EC A1E           C=C-1  WP              Dec exp, Negative?
 865 0D7EF 5BE           GONC   RED360          Nope, continue reduce
 866 0D7F2 A92           C=0    WP              Set extended EXP zero
 867                ************************************************************
 868
 869
 870
 871                ************************************************************
 872                *   Argument align and constant generation for reduce 90   *
 873 0D7F5 128    TRG160 CROEX                  Mant ---> (A,C)
 874 0D7F8 7003          GOSUB  DBLPI4          fetch dbl pi/4  or 45
 875 0D7FC 7833          GOSUB  TWO*            pi/2  or  90
 876 0D800 869           ?ST=0  =sRAD           Deg mode?
 877 0D803 A0            GOYES  RED90           If yes, skip alignment shifts
 878 0D805 7653          GOSUB  SHFRBD          Shift PI/2 right
 879 0D809 7443          GOSUB  SHFRAC          Shift Theta right
 880                ************************************************************
 881
 882
 883
 884                ************************************************************
 885                *          31-dig Trig reduction (90 or pi/2)              *
 886 0D80D 7CC2  RED90  GOSUB  DBLSUB          REDUCE Q BY 90 OR PI/2. BORROW?
 887 0D811 451           GOC    TRG180          Yes, go TRG180
 888 0D814 7393          GOSUB  FLIP11          TAN(a+90)=-COT(a)
 889 0D818 7173          GOSUB  FLIP8           SIN(a+90)=COS(a)
 890 0D81C 40F           GOC    RED90           COS(a+90)=-SIN(a)
 891 0D81F 7973          GOSUB  FLIP10          Toggle Sign flag
 892 0D823 69EF          GOTO   RED90
 893                ************************************************************
 894
 895 0D827 128   TRG180 CROEX                  Fetch exp to C
 896 0D82A 869   TRG181 ?ST=0  =sRAD
 897 0D82D 71            GOYES  TRG270          If DEG Mode, go TRG270
 898 0D82F 91E           ?C#0   WP              EXP NONZERO?
 899 0D832 F0            GOYES  TRG260          YES (Arg small, didn't reduce)
 900 0D834 128           CROEX
 901 0D837 7B03          GOSUB  SHFLAC          Shift theta left
 902 0D83B 128           CROEX
 903 0D83E 550           GONC   TRG270          BET.
 904 0D841 B16   TRG260 C=C+1  WP              Bump exp
 905
 906 0D844 90E   TRG270 ?C#0   P               C:EXP,  R0:Low Theta
 907 0D847 24            GOYES  TRG279          If Neg EXP, go TRG279
 908 0D849 128           CROEX
 909 0D84C 7CA2          GOSUB  DBLPI4          Get pi/4 or 45
 910 0D850 7982          GOSUB  DBLSUB          Theta>=45 ?
 911 0D854 401           GOC    TRG275          No, use theta
 912 0D857 AFC           ABEX   W               Yes, 90-theta
 913 0D85A AFF           CDEX   W               TAN(90-Q)=COT(Q)
 914 0D85D 7C72          GOSUB  DBLSUB          SIN(90-Q)=COS(Q)
 915 0D861 7823          GOSUB  FLIP8           COS(90-Q)=SIN(Q)
 916
```

```
917
918             ************************************************************
919             *     Normalize reduced arg if necessary [15-dig result]   *
920 0D865 2E    TRG275 P=     14
921 0D867 128          CROEX
922 0D86A AF5          B=C    W              Exp to B
923 0D86D 128          CROEX
924 0D870 97C          ?A#0   W              High theta zero?
925 0D873 70           GOYES  NORM           If non-zero, go normalize
926 0D875 97A          ?C=0   W              Low order theta zero also?
927 0D878 41           GOYES  TRG280         If yes, skip normalize
928
929 0D87A 90C   NORM   ?A#0   P              Lead digit non-zero?
930 0D87D F0           GOYES  TRG280         If so, done normalize
931 0D87F 73C2         GOSUB  SHFLAC         double word left shift
932 0D883 CD           B=B-1  A              Decrement exp
933 0D885 64FF         GOTO   NORM           Do again
934             ************************************************************
935
936
937
938             ************************************************************
939             *     Exactness: RAD Mode 0, or TRG(0),SIN(30),TAN(45)     *
940 0D889 AF5   TRG279 B=C    W
941 0D88C       TRG280                                  A:Mant  B:Exp
942 0D88C AFC          ABEX   W
943 0D88F 879          ?ST=1  =sRAD
944 0D892 B2           GOYES  TRG290         If RAD, only zero EXACT.
945 0D894 979          ?B=0   W
946 0D897 32           GOYES  TRGEX          If reduced arg zero, EXACT.
947 0D899 8AC          ?A#0   A
948 0D89C 12           GOYES  TRG290         If Exp#0, can't be 30, 45.
949 0D89E AF2          C=0    W
950 0D8A1 2D           P=     13
951 0D8A3 3154         LCHEX  45
952 0D8A7 866          ?ST=0  sTAN
953 0D8AA B0           GOYES  TRGTST         If Tan, check 45.
954 0D8AC 868          ?ST=0  sINVRT
955 0D8AF E0           GOYES  TRG290         If Cos, inexact.
956 0D8B1 3103         LCHEX  30             Else Sin, check 30.
957 0D8B5 975   TRGTST ?B#C   W
958 0D8B8 50           GOYES  TRG290
959 0D8BA 847   TRGEX  ST=0   sIX            Exact result (S7=0)
960             ************************************************************
961
962
963
964             ************************************************************
965             *         If Degree Mode, convert arg to Radians           *
966 0D8BD CC    TRG290 A=A-1  A              DEC EXP
967 0D8BF 879          ?ST=1  =sRAD          RAD MODE?
968 0D8C2 31           GOYES  TRG300         YES
969 0D8C4 27           P=     7
970 0D8C6 79D1         GOSUB  GETCON         Fetch PI/180
971 0D8CA AF7          D=C    W
```

```
 972 0D8CD D2          C=0    A             Make 15-digit form
 973 0D8CF 8E00        GOSUBL =MULTF        (PI/180)*Theta
           00
 974           ************************************************************
 975
 976
 977
 978 0D8D5 D6    TRG300 C=A    A             Special case: EXP=-1
 979 0D8D7 E6           C=C+1  A             Inc exp, carry?
 980 0D8D9 5A0          GONC   TRG310        No, Go see how small
 981 0D8DC BF1          BSL    W
 982 0D8DF 2E           P=     14            Yes, shift Mantissa and
 983 0D8E1 492          GOC    TRG330        (BET) go pseudo-divide (p=14)
 984
 985 0D8E4 D2    TRG310 C=0    A             EXP<=-2; How small is it?
 986 0D8E6 20           P=     0
 987 0D8E8 308          LCHEX  8
 988 0D8EB C2           C=C+A  A             -8<=EXP ?
 989 0D8ED 411          GOC    TRG318        Yes, go TRG318
 990 0D8F0 868          ?ST=0  sINVRT        No, very small.  1/TAN ?
 991 0D8F3 80           GOYES  TRG314        No, ARG=Answer
 992 0D8F5 8E00         GOSUBL =1/X15        Yes, Invert ARG
           00
 993 0D8FB 6BA0  TRG314 GOTO   TRG430
 994
 995 0D8FF AE6   TRG318 C=A    B
 996 0D902 25           P=     5             INDEX: J=EXP+16
 997 0D904 809          C+P+1
 998 0D907 80D0         P=C    0
 999
1000           ************************************************************
1001           *                    PSEUDO-DIVIDE                         *
1002 0D90B AF0   TRG330 A=0    W             PSEUDO-DIVIDE (A=PQUOT)
1003 0D90E 7191  TRG340 GOSUB  GETCON        Fetch PHI=ATAN(10**-i).
1004 0D912 550          GONC   TRG800        BET
1005 0D915 B44   TRG810 A=A+1  S             Bump current PQUOT
1006 0D918 B71   TRG800 B=B-C  W             Theta=Theta-PHI. Borrow?
1007 0D91B 59F          GONC   TRG810        Nope, do again
1008 0D91E A71          B=B+C  W             Restore Theta
1009 0D921 0D           P=P-1                Decrement INDEX J
1010 0D923 BF4          ASR    W             Shift Previous PQUOT'S
1011 0D926 BF1          BSL    W             Align Theta
1012 0D929 887          ?P#    7             J=7? (Done PSUEDO-DIVIDE?)
1013 0D92C 2E           GOYES  TRG340        Nope
1014 0D92E BF5          BSR    W
1015 0D931 BF5          BSR    W             Form Y(8)
1016 0D934 AF6          C=A    W
1017 0D937 2F           P=     15
1018 0D939 3177         LCHEX  77
1019 0D93D AFA          A=C    W             A = [1,PQUOT,1]    (i=7)
1020 0D940 AF3          D=0    W
1021 0D943 2E           P=     14
1022 0D945 B07          D=D+1  P             D=X(8)=1
1023 0D948 AF9          C=B    W             B=C=Y(8)
1024 0D94B 5A1          GONC   TRG370        BET - Go unravel PQUOT
```

```
1025            ****************************************************************
1026
1027
1028
1029            ****************************************************************
1030            *                      UNRAVEL PQUOTE                          *
1031 0D94E B96   TRG350 CSR    WP              Form (10**-2i)Y
1032 0D951 B96          CSR    WP
1033 0D954 A4E   TRG360 C=C-1  S               j=j-1.  Borrow?
1034 0D957 56F          GONC   TRG350          Nope, keep shifting
1035 0D95A AC2          C=0    S
1036 0D95D AFF          CDEX   W               D=(10**-2i)Y. C=X
1037 0D960 B7F          D=C-D  W               X'=X-(10**-2i)Y
1038 0D963 A79          C=C+B  W               Y'=X+Y
1039 0D966 AF5   TRG370 B=C    W               B=Y=Y'
1040 0D969 AC6          C=A    S               j=i (shift counter)
1041 0D96C A0C          A=A-1  P               PQUOT=PQUOT-1. Borrow?
1042 0D96F 54E          GONC   TRG360          Nope, continue (same i)
1043 0D972 BD0          ASL    M
1044 0D975 958          ?A=0   M               Rest of PQUOT=0?
1045 0D978 21           GOYES  TRG400          Yep, Done- C=Y, D=X
1046 0D97A A4C          A=A-1  S               Nope, decrement i
1047 0D97D A6C          A=A-1  B               decrement exp
1048 0D980 AC2          C=0    S
1049 0D983 BF6          CSR    W               Y=10*Y
1050 0D986 6FDF         GOTO   TRG370
1051            ****************************************************************
1052
1053
1054
1055 0D98A AC0   TRG400 A=0    S
1056 0D98D AC2          C=0    S               Clear junk in signs
1057 0D990 A7F          D=D-1  W               PROTECT FROM 10 DIVIDE
1058 0D993 F8           A=-A   A               EXP(Y)=-i
1059 0D995 D1           B=0    A               EXP(X)=0
1060 0D997 868          ?ST=0  sINVRT          Y/X?
1061 0D99A 70           GOYES  TRG420          YES
1062 0D99C F8           A=-A   A               COMP EXP
1063 0D99E AFF          CDEX   W               X/Y
1064 0D9A1 8E00 TRG420  GOSUBL =DIVFCD         Z=Y/X or X/Y
          00
1065 0D9A7 876   TRG430 ?ST=1  sTAN            SIN or COS ?
1066 0D9AA 51           GOYES  TRG440          Yep, go TRG440
1067 0D9AC 831          ?XM=0                  No zero divide?
1068 0D9AF 14           GOYES  TRG500          Yep, go TRG500
1069 0D9B1 BCC          A=-A-1 S               Fudge sign (toggle) for INF
1070 0D9B4 20           P=     0
1071 0D9B6 3100         LC(2)  =eTNINF         MSG="Tan=INF"
1072 0D9BA 20           P=     =DZP            Restore err code
1073 0D9BC 5C3          GONC   TRG515          BET. Go set sgn of INF
1074
1075 0D9BF 119   TRG440 C=R1                   Fetch flag
1076 0D9C2 94A          ?C=0    S              COSINE?
1077 0D9C5 B0           GOYES   TRG450         NO.
1078 0D9C7 831          ?XM=0                  TAN=INF?
```

```
1079 0D9CA 60           GOYES   TRG450          NO.
1080 0D9CC 7CC1         GOSUB   FLIP10          Toggle sign (COS(90), etc.)
1081 0D9D0 7E3D  TRG450 GOSUB   STAB1X          Z=TAN OR COT--->R0,R1
1082 0D9D4 7B91         GOSUB   RCCD1X
1083 0D9D8 8E00         GOSUBL  =MP2-15         FORM Z**2
           00
1084 0D9DE 8E00         GOSUBL  =ADDONE         FORM 1+Z**2.  Not 0, ==> Round
           00
1085                                                    mode not important.
1086 0D9E4 8E00         GOSUBL  =SQR15          FORM SQRT(1+Z**2)
           00
1087 0D9EA 8E00         GOSUBL  =1/X15          FORM 1/SQRT(1+Z**2)
           00
1088
1089 0D9F0 7214  TRG500 GOSUB   FUDGE           Make perfect if exact.
1090 0D9F4 876   TRG510 ?ST=1   sTAN            SIN or COS ?
1091 0D9F7 C0           GOYES   TRG530          Yep, go TRG530
1092 0D9F9 86B   TRG515 ?ST=0   sSGNT           Negate TAN ?
1093 0D9FC 00           RTNYES                  No
1094 0D9FE BCC   TRG520 A=-A-1  S               Complement sign
1095 0DA01 01           RTN
1096 0DA03 86A   TRG530 ?ST=0   sSGN            Negate result?
1097 0DA06 00           RTNYES                  No
1098 0DA08 55F          GONC    TRG520          Yes (BET)
1099
1100          ****************************************************************
1101
1102
1103 0DA0B 8C00 =SWAPXY GOLONG =XYEX              JT%MTH    SWAP X & Y
           00
1104                                            (moved here to pack code.  BS)
1105
```

```
1106                    EJECT
1107                ********************************************************************
1108                ********************************************************************
1109                ** Name:(S) TRC90   -  Table of numeric constants
1110                **
1111                ** Category: MATH
1112                **
1113                ** Purpose:  Constants used by the trig routines.
1114                **
1115                ** Entry:    Values are accessed by a call to GETCON with a
1116                **           select code in P (See GETCON, GETVAL).
1117                **
1118                ** History:
1119                **
1120                **     Date      Programmer             Modification
1121                **   --------   ----------   ---------------------------------
1122                **   11/02/83   SB           Documented
1123                ********************************************************************
1124                ********************************************************************
1125 0DA11 0261 =TRC90  NIBHEX 0261194256866990  ATAN(10^-1)  14
           1942
           5686
           6990
1126 0DA21 4256         NIBHEX 4256668666699990  ATAN(10^-2)  13
           6686
           6669
           9990
1127 0DA31 7686         NIBHEX 7686666669999990  ATAN(10^-3)  12
           6666
           6999
           9990
1128 0DA41 7666         NIBHEX 7666666999999990  ATAN(10^-4)  11
           6669
           9999
           9990
1129 0DA51 7666         NIBHEX 7666699999999990  ATAN(10^-5)  10
           6999
           9999
           9990
1130 0DA61 7669         NIBHEX 7669999999999990  ATAN(10^-6)   9
           9999
           9999
           9990
1131 0DA71 7999         NIBHEX 7999999999999990  ATAN(10^-7)   8
           9999
           9999
           9990
1132 0DA81 3349         NIBHEX 3349915292354710  PI/180        7
           9152
           9235
           4710
1133 0DA91 8447         NIBHEX 8447933618935870  PI/4          6
           9336
           1893
           5870
```

1134

```
1135                    EJECT
1136 0DAA1 26   =PI/4   P=     6              Use GETCON to fetch PI/4
1137            ****************************************************************
1138            ****************************************************************
1139            **
1140            ** Name:(S) GETCON  -  Get constants from table
1141            ** Name:(S) GETVAL  -  Get constants from table
1142            ** Name:(S) PI/4    -  Fetch Pi/4 from table
1143            **
1144            ** Category:   MTHUTL
1145            **
1146            ** Purpose:    Access numeric constants stored in table.
1147            **
1148            ** Entry:      Table index in P (Selects desired constant).
1149            **
1150            ** Exit:       Constant selected in C
1151            **
1152            ** Alters (INC): C,D[A]
1153            **
1154            ** Stk lvls:   0
1155            **
1156            ** NOTE:
1157            **      Presently used only for constant table starting at
1158            **      label TRC90. However by entering at label GETVAL, this
1159            **      code can be used to access constants stored in other
1160            **      tables.  The 1st constant corresponds to P=14, the 2nd
1161            **      to P=13, etc.
1162            **
1163            ** Algorithm:  Value of P determines offset from table start.
1164            **
1165            ** History:
1166            **
1167            **     Date     Programmer            Modification
1168            **   --------   ----------   ---------------------------------
1169            **    6/07/82   SB           Documented
1170            **    9/30/82   SB           Use of D[A] instead of stack.
1171            **   01/06/83   SB           New entry: PI/4
1172            **   02/07/83   SB           Move =PI/4 above header-Cosmetic
1173            **                           change only.
1174            ****************************************************************
1175            ****************************************************************
1176 0DAA3 D2   =GETCON C=0    A
1177 0DAA5 80F1        CPEX   1
1178 0DAA9 D7          D=C    A               D[A]=[000P0] (P in Nib 1)
1179 0DAAB 341F        LC(5)  (=TRC90)+#E0    Address of P=0  constant
          ADO
1180 0DAB2 04   =GETVAL SETHEX
1181 0DAB4 EB          C=C-D  A               Address of desired constant
1182 0DAB6 05          SETDEC
1183 0DAB8 136         CDOEX
1184 0DABB DF          CDEX   A               Save D0,  Fetch back P
1185 0DABD 80D1        P=C    1               Restore P
1186 0DAC1 1567        C=DATO W               Fetch constant
1187 0DAC5 DF          CDEX   A               Cleanup: Restore D0
1188 0DAC7 134         D0=C
```

```
1189 0DACA DB          C=D    A
1190 0DACC 03          RTNCC
1191
```

```
1192                    EJECT
1193                ********************************************************************
1194                ********************************************************************
1195                ** Name:(S) MAKE1   -  Make 12-dig 1 in C and compare with B.
1196                **
1197                ** Category: MTHUTL
1198                **
1199                ** Purpose:  Make 12-dig 1.0 in C and test against value in B
1200                **
1201                ** Entry:    DEC Mode
1202                **
1203                ** Exit:     C: [0100000000000000], P=14; CARRY Set iff B=C
1204                **
1205                ** Alters:   C,P,CARRY
1206                **
1207                ** Calls:    Nothing
1208                **
1209                ** Stack Levels: 0
1210                **
1211                ** History:
1212                **
1213                **     Date     Programmer            Modification
1214                **   --------   ----------   ---------------------------------
1215                **   11/02/83   SB           Documented
1216                ********************************************************************
1217                ********************************************************************
1218 0DACE AF2  =MAKE1  C=0    W
1219 0DAD1 2E          P=     14
1220 0DAD3 B06         C=C+1  P
1221 0DAD6 971         ?B=C   W
1222 0DAD9 00          RTNYES
1223 0DADB 01          RTN
1224
```

```
1225                    EJECT
1226                ********************************************************************
1227                ********************************************************************
1228                **
1229                ** Name:(S) DBLSUB  -  Double Precision Subtract
1230                **
1231                ** Category:   MTHUTL
1232                **
1233                ** Purpose:    Dbl Precision subtract (used in TRIG Reduction).
1234                **
1235                ** Entry:      A&C:Y,   B&D:X  31-digit positive fixed point
1236                **             values. First 15 high order digits are in A & B.
1237                **             Notation: XH=high order word of X.
1238                **
1239                ** Exit:       A&C:Z
1240                **             Carry Clear: Z=Y-X
1241                **             Carry Set  : Z=Y  (In this case Y<X)
1242                **
1243                **                          A     B     C     D
1244                **                (ENTRY)  YH    XH    YL    XL
1245                **                (EXIT)   ZH    XH    ZL    XL
1246                **
1247                ** Alters (INC): A,C,Carry
1248                **
1249                ** Stk lvls:   0
1250                **
1251                ** History:
1252                **
1253                **     Date      Programmer             Modification
1254                **   --------   ----------   ---------------------------------
1255                **    6/07/82   SB           Documented
1256                ********************************************************************
1257                ********************************************************************
1258 0DADD B70  =DBLSUB A=A-B  W              High order subtract
1259 0DAE0 461         GOC    DBL4
1260 0DAE3 B7B         C=C-D  W              Low order subtract
1261 0DAE6 440         GOC    DBL3           Borrow needed?
1262 0DAE9 03          RTNCC                 Successful subtract
1263 0DAEB A7C   DBL3  A=A-1  W              Take care of borrow
1264 0DAEE 500         RTNNC                 Successful subtract
1265 0DAF1 B74         A=A+1  W              Un-do borrow
1266 0DAF4 A7B         C=C+D  W              Un-do Low order subtract
1267 0DAF7 A70   DBL4  A=A+B  W              Un-do high order subtract
1268 0DAFA 02          RTNSC                 Unsuccessful subtract
1269
```

```
1270                    EJECT
1271                ****************************************************************
1272                ****************************************************************
1273                **
1274                ** Name:(S) DBLPI4  -  Generate 31-digit PI/4 or 45
1275                **
1276                ** Category:   MTHUTL
1277                **
1278                ** Purpose:    Generate 31-digit value PI/4 -or- 45
1279                **
1280                ** Entry:      sRAD Status bit (sRAD=1 ==> PI/4,  ELSE 45)
1281                **
1282                ** Exit:       Value in [B,D],  P=5.
1283                **
1284                ** Calls:      PI/4
1285                **
1286                ** Alters (INC): B,D,P,Carry
1287                **
1288                ** Stk lvls:   1
1289                **
1290                ** History:
1291                **
1292                **     Date     Programmer            Modification
1293                **   --------   ----------   ---------------------------------
1294                **    6/07/82   SB           Documented
1295                **   10/05/82   SB           Code Pack
1296                **   10/06/82   SB           Code Pack - Eliminate call GETCON
1297                **   01/06/83   SB           Fix header, Pack by moving the
1298                **                           entry PI/4 to before GETCON.
1299                ****************************************************************
1300                ****************************************************************
1301 0DAFC AF5  =DBLPI4 B=C    W                    Save (C)
1302 0DAFF 869         ?ST=0   =sRAD
1303 0DB02 72          GOYES   DBL45                If DEG Mode, go create 45
1304 0DB04 799F        GOSUB   PI/4                 High pi/4
1305 0DB08 AF7         D=C     W
1306 0DB0B 20          P=      0
1307 0DB0D 3F99        LCHEX   3096156608458199     Low  pi/4
           1854
           8066
           5169
           03
1308 0DB1F AFF         CDEX    W
1309 0DB22 25    EGRESS P=      5
1310 0DB24 AFD         BCEX    W                    Restore (C)
1311 0DB27 01          RTN
1312 0DB29 AF2   DBL45  C=0     W
1313 0DB2C AF3         D=0     W
1314 0DB2F 2D          P=      13
1315 0DB31 3154        LCHEX   45
1316 0DB35 4CE         GOC     EGRESS               BET. Go exit
1317
1318
1319
```

```
1320                    EJECT
1321                ********************************************************************
1322                ********************************************************************
1323                **
1324                ** Name:(S) TWO*    -  Double Precision Doubler
1325                **
1326                ** Category:   MTHUTL
1327                **
1328                ** Purpose:    Dbl Precision doubler
1329                **
1330                ** Entry:      B&D:X (B:XH, D:XL)
1331                **
1332                ** Exit:       B&D: 2*X
1333                **
1334                ** Alters (INC): B,D,Carry
1335                ********************************************************************
1336                ********************************************************************
1337 0DB38 A75    =TWO*   B=B+B  W              Double the value in (B,D)
1338 0DB3B A77            D=D+D  W              Low order add
1339 0DB3E 500            RTNNC                Exit if no carry from low add
1340 0DB41 B75            B=B+1  W              Carry reflected to high order
1341 0DB44 01             RTN
1342
1343                ********************************************************************
1344                ********************************************************************
1345                **
1346                ** Name:(S) SHFLAC  -  Double Precision Shift Left
1347                ** Name:(S) SHFRAC  -  Double Precision Shift Right
1348                **
1349                ** Category:   MTHUTL
1350                **
1351                ** Purpose:    Dbl Precision (Fixed Point) shifts
1352                **
1353                ** Entry:      A&C:X (A:XH, C:XL)
1354                **
1355                ** Exit:       A&C:10*X  (or X/10)
1356                **
1357                ** Alters (INC): A,C,(SHFRAC Only - SB)
1358                ********************************************************************
1359                ********************************************************************
1360 0DB46 ACA    =SHFLAC A=C    S              Capture MSD of theta low.
1361 0DB49 810            ASLC                 Put it as LSD theta high.
1362 0DB4C BF2            CSL    W             Shift theta low left.
1363 0DB4F 01             RTN
1364
1365 0DB51 BF6    =SHFRAC CSR    W             Lose LSD of theta low.
1366 0DB54 814            ASRC                 LSD of theta high to sign.
1367 0DB57 AC6            C=A    S             It becomes MSD of theta low.
1368 0DB5A AC0            A=0    S             Shift 0 into theta high.
1369 0DB5D 01             RTN
1370
```

```
1371                    EJECT
1372                ****************************************************************
1373                ****************************************************************
1374                **
1375                ** Name:(S) SHFRBD  -  Double Precision Right Shift
1376                **
1377                ** Category:   MTHUTL
1378                **
1379                ** Purpose:    Dbl Precision (Fixed Point) right shift
1380                **
1381                ** Entry:      B&D:X (B:XH, D:XL)
1382                **
1383                ** Exit:       B&D:X/10
1384                **
1385                ** Alters (INC): B,D,SB
1386                ****************************************************************
1387                ****************************************************************
1388 0DB5F BF7    =SHFRBD DSR    W              Lose LSD of D.
1389 0DB62 815          BSRC                   Rotate LSD of B to B[S].
1390 0DB65 ACD          CBEX   S               Make accessible to D.
1391 0DB68 AC7          D=C    S               Move it to MSD of D.
1392 0DB6B ACD          CBEX   S               Restore C[S].
1393 0DB6E AC1          B=0    S               Clear MSD of B.
1394 0DB71 01           RTN
1395
1396
1397                ********************************************************
1398 0DB73 6000 =RCCD1X GOTO   =RCCD1           JT%MTH    FETCH Y FROM R0,R1
1399                ********************************************************
1400
1401                ****************************************************************
1402                ****************************************************************
1403                **
1404                ** Name:(S) PI/2    -  Generate PI/2
1405                ** Name:(S) PI/2D   -  Generate signed PI/2
1406                **
1407                ** Category:   MTHUTL
1408                **
1409                ** Purpose:    Generate Pi/2 (15-Digit form)
1410                **
1411                ** Exit:       CD:  1.57079632679490
1412                **
1413                ** Calls:      PI/4
1414                **
1415                ** Alters (INC): C,D,P,Carry
1416                **
1417                ** Stk lvls:   1
1418                ****************************************************************
1419                ****************************************************************
1420 0DB77 AC3  =PI/2   D=0    S              SGN="+"
1421 0DB7A 732F =PI/2D  GOSUB  PI/4           0785398163397448
1422 0DB7E A76          C=C+C  W              1570796326794896
1423 0DB81 BF6          CSR    W              0157079632679489
1424 0DB84 E6           C=C+1  A              0157079632679490 (MAKE BETTER)
1425 0DB86 AFF          CDEX   W
```

```
1426 0DB89 D2          C=0    A            EXP=0
1427 0DB8B 01          RTN
1428
```

```
1429                    EJECT
1430                ********************************************************************
1431                ********************************************************************
1432                **
1433                ** Name:(S) FLIP8   -  Toggle status bits
1434                ** Name:(S) FLIP10  -  Toggle status bits
1435                ** Name:(S) FLIP11  -  Toggle status bits
1436                **
1437                ** Category:   MTHUTL
1438                **
1439                ** Purpose:    Toggle Status bits
1440                **
1441                ** Exit:       Toggled status, Carry set if new status = 0.
1442                **
1443                ** Alters (INC): Selected Status bit, Carry.
1444                ********************************************************************
1445                ********************************************************************
1446 0DB8D 878  =FLIP8  ?ST=1   sINVRT
1447 0DB90 70           GOYES   TOG8
1448 0DB92 858          ST=1    sINVRT
1449 0DB95 03           RTNCC                   Clr Carry:  0-->1
1450 0DB97 848   TOG8   ST=0    sINVRT
1451 0DB9A 02           RTNSC                   Set Carry:  1-->0
1452
1453 0DB9C 87A  =FLIP10 ?ST=1   sSGN
1454 0DB9F 70           GOYES   TOG10
1455 0DBA1 85A          ST=1    sSGN
1456 0DBA4 03           RTNCC                   Clr Carry:  0-->1
1457 0DBA6 84A   TOG10  ST=0    sSGN
1458 0DBA9 02           RTNSC                   Set Carry:  1-->0
1459
1460 0DBAB 87B  =FLIP11 ?ST=1   sSGNT
1461 0DBAE 70           GOYES   TOG11
1462 0DBB0 85B          ST=1    sSGNT
1463 0DBB3 03           RTNCC                   Clr Carry:  0-->1
1464 0DBB5 84B   TOG11  ST=0    sSGNT
1465 0DBB8 02           RTNSC                   Set Carry:  1-->0
1466
1467
```

```
1468                    EJECT
1469            ****************************************************************
1470            ****************************************************************
1471            **
1472            ** Name:(S) ASIN12  -  ArcSin Inv Trig (12-dig argument)
1473            ** Name:(S) ACOS12  -  ArcCos Inv Trig (12-dig argument)
1474            ** Name:    ATAN12  -  ArcTan Inv Trig (12-dig argument)
1475            ** Name:(S) ASIN15  -  ArcSin Inv Trig (15-dig argument)
1476            ** Name:(S) ACOS15  -  ArcCos Inv Trig (15-dig argument)
1477            ** Name:(S) ATAN15  -  ArcTan Inv Trig (15-dig argument)
1478            ** Name:(S) BRT30   -  Inv Trig, defined by status
1479            ** Name:(S) BRTF    -  Inv Trig, finite arg, defined by status
1480            **
1481            ** Category: MATH
1482            **
1483            ** Purpose:  ARCSINE, ARCCOSINE, ARCTANGENT
1484            **
1485            ** Entry:    ASIN12,ACOS12,ATAN12 - Stnd. Math, 12-dig arg'ts
1486            **           ASIN15,ACOS15,ATAN15 - Stnd. Math, 15-dig arg'ts
1487            **
1488            **                All entries assume angle mode encoded in
1489            **                status bit =sRAD (set=RAD Mode).
1490            **
1491            ** Exit:     Standard math (15-digit result in A&B)
1492            **
1493            ** Calls:    SPLTA,AFIN,=INVNaN,PI/2,SWAPXY,STAB1X,
1494            **           =STAB2,=ADDONE,=EXAB1,=SUBONE,RCCD1X,=MULTF,
1495            **           =SQR17,=RCCD2,=X/Y15,=1/X15,FLIP8,GETCON,
1496            **           =DIV120,=SHF10,=AD15s,=DIV100,FUDGE
1497            **
1498            ** Alters (INC): A,B,C,D,R0,R1,R2,R3,P,XM,SB,sIX,sCOMP,
1499            **                sATAN,sSGN,s+PI/2
1500            ** Stk lvls:  2
1501            **
1502            ** Algorithm:
1503            **
1504            ** sIX    (7) : If set, result may be inexact    (INEXACT flag)
1505            ** sCOMP  (8) : If set, need complementary angle (COMP flag)
1506            ** sATAN  (6) : If set, need ATAN                (ATAN flag)
1507            ** sSGN   (10): If set, negate result           (SGN flag)
1508            ** s+PI/2 (11): If set, need add PI/2           (Add PI/2)
1509            **
1510            ** History:
1511            **
1512            **     Date     Programmer            Modification
1513            **   --------   ----------   ---------------------------------
1514            **    6/07/82   SB           Documented
1515            **   10/06/82   SB           Code Pack:  Eliminate proj mode
1516            ****************************************************************
1517            ****************************************************************
1518 0DBBA 784B =ATAN12 GOSUB  SPLTA
1519 0DBBE 848  =ATAN15 ST=0   sCOMP          ATAN: sCOMP=0
1520 0DBC1 846          ST=0   sATAN                sATAN=0
1521 0DBC4 6810         GOTO   BRT20
1522
```

```
1523 0DBC8 7A3B =ASIN12 GOSUB  SPLTA
1524 0DBCC 848  =ASIN15 ST=0   sCOMP          ASIN: sCOMP=0
1525 0DBCF 6A00         GOTO   BRT10                sATAN=1
1526
1527 0DBD3 7F2B =ACOS12 GOSUB  SPLTA
1528 0DBD7 858  =ACOS15 ST=1   sCOMP          ACOS: sCOMP=1 (ACOS=PI/2-ASIN)
1529 0DBDA 856   BRT10  ST=1   sATAN                sATAN=1
1530
1531
1532 0DBDD 821   BRT20  XM=0                   INIT: (NO EXCPT'NS)
1533 0DBE0 847         ST=0   sIX                    (EXACT)
1534 0DBE3 84A  =BRT30 ST=0   sSGN                  sSGN=0    [+]
1535 0DBE6 84B         ST=0   s+PI/2                s+PI/2=0  [No add pi/2]
1536 0DBE9 731B        GOSUB  AFIN           Finite ?
1537 0DBED 572         GONC   BRTF           Yes.
1538 0DBF0 96C         ?A#0   B              Not finite.  NaN?
1539 0DBF3 00          RTNYES                Yes, send it back.
1540 0DBF5 866         ?ST=0  sATAN          INF.  ATAN?
1541 0DBF8 60          GOYES  BRT40           Yes, go BRT40
1542
1543 0DBFA 6E4B  OOPS5 GOTO   IVARG          Msg="Invalid Arg"
1544
1545 0DBFE 757F  BRT40 GOSUB  PI/2            Fetch pi/2
1546 0DC02 750E        GOSUB  SWAPXY
1547 0DC06 ACA         A=C    S              Sign of ARG*pi/2
1548 0DC09 869         ?ST=0  =sRAD          DEG Mode ?
1549 0DC0C 50          GOYES  BRT60           Yes
1550 0DC0E 857         ST=1   sIX            No, so Inexact
1551 0DC11 69C1  BRT60 GOTO   BRT420         ATAN(INF)=PI/2  (+/-)
1552
1553
1554 0DC15       =BRTF                          *** FINITE ARGUMENT
1555 0DC15 AC1         B=0    S              Ensure zero field
1556 0DC18 879         ?ST=1  =sRAD
1557 0DC1B A2          GOYES  BRTRAD         If RAD Mode, go BRTRAD
1558 0DC1D 979         ?B=0   W              If ARG zero, Exact. go BRT110
1559 0DC20 64          GOYES  BRT110          ...
1560 0DC22 78AE        GOSUB  MAKE1           ...
1561 0DC26 570         GONC   BRT80           ...
1562 0DC29 8A8         ?A=0   A               ...
1563 0DC2C A3          GOYES  BRT110         If ARG 1, Exact. go BRT110
1564 0DC2E 866   BRT80 ?ST=0  sATAN
1565 0DC31 23          GOYES  BRTNEX         If ATAN, Inexact. Go BRTNEX
1566 0DC33 305         LCHEX  5               ...
1567 0DC36 975         ?B#C   W               ...
1568 0DC39 A2          GOYES  BRTNEX          ...  ARG= (+/-)0.5 ?
1569 0DC3B CE          C=C-1  A               ...
1570 0DC3D 8A2         ?A=C   A               ...
1571 0DC40 62          GOYES  BRT110          Yes, Exact.
1572 0DC42 502         GONC   BRTNEX          BET.
1573
1574 0DC45 868   BRTRAD ?ST=0  sCOMP
1575 0DC48 61          GOYES  BRT100         If ASIN or ATAN, go BRT100
1576 0DC4A 94C         ?A#0   S               ...
1577 0DC4D 61          GOYES  BRTNEX          ...
```

```
1578 0DC4F 8AC          ?A#0   A            ...
1579 0DC52 11           GOYES  BRTNEX        ...
1580 0DC54 767E         GOSUB  MAKE1         ...  ARG=+1 ?
1581 0DC58 5A0          GONC   BRTNEX        ...
1582 0DC5B 4A0          GOC    BRT110        Yes. (BET)  ACOS(1)
1583 0DC5E 979   BRT100 ?B=0   W
1584 0DC61 50           GOYES  BRT110
1585 0DC63 857   BRTNEX ST=1    sIX          Inexact.
1586
1587
1588 0DC66       BRT110                        *** EXACTNESS ESTABLISHED
1589 0DC66 948          ?A=0   S            SIGN(arg) = "+" ?
1590 0DC69 31           GOYES  BRT130        YES
1591 0DC6B 85A          ST=1   sSGN          SET NEG ANS FLAG
1592 0DC6E 868          ?ST=0  sCOMP         ASIN,ATAN?
1593 0DC71 B0           GOYES  BRT130        Yes. Go BRT130
1594 0DC73 848          ST=0   sCOMP
1595 0DC76 85B          ST=1   s+PI/2        ACOS(-X)=PI/2 + ASIN(X)
1596 0DC79 84A          ST=0   sSGN          Reset Neg Ans flag
1597
1598 0DC7C 97D   BRT130 ?B#0   W
1599 0DC7F 60           GOYES  BRT135
1600 0DC81 6F21         GOTO   BRT400        If Arg=0, go BRT400
1601 0DC85 D6    BRT135 C=A     A
1602 0DC87 C6           C=C+C  A
1603 0DC89 5A5          GONC   BRT150        If  |Arg| >= 1   go BRT150
1604
1605 0DC8C 866   BRT140 ?ST=0  sATAN         |Arg|<1.
1606 0DC8F 15           GOYES  BRT145        If ATAN, go BRT145
1607
1608 0DC91 7D7A         GOSUB  STAB1X        ACOS or ASIN. Use Formula:
1609 0DC95 8E00         GOSUBL =STAB2        ASIN(X)=ATAN(X/SQRT(1-X^2))
          00
1610 0DC9B 8E00         GOSUBL =ADDONE       ARG+1  (See note below)
          00
1611 0DCA1 8E00         GOSUBL =EXAB1
          00
1612 0DCA7 8E00         GOSUBL =SUBONE       ARG-1  (See note below)
          00
1613 0DCAD 72CE         GOSUB  RCCD1X
1614 0DCB1 8E00         GOSUBL =MULTF        ARG**2-1
          00
1615 0DCB7 AC0          A=0    S             1-ARG**2
1616 0DCBA 8E00         GOSUBL =SQR17        SQRT(1-ARG**2)
          00
1617 0DCC0 8E00         GOSUBL =RCCD2
          00
1618 0DCC6 8E00         GOSUBL =X/Y15        Z=ARG/SQRT(1-ARG**2)
          00
1619 0DCCC AC0          A=0    S             Set positive.  NOTE: Because
1620                                               of this, round mode not
1621                                               important in the calls
1622                                               to ADDONE & SUBONE.
1623 0DCCF D6           C=A    A
1624 0DCD1 C6           C=C+C  A             Z<1 ?
```

```
1625 0DCD3 4C0          GOC    BRT145          Yes, go BRT145
1626 0DCD6 8E00 BRT144  GOSUBL =1/X15          Z>=1. ATAN(Z)=PI/2-ATAN(1/Z)
          00
1627 0DCDC 7DAE         GOSUB  FLIP8
1628 0DCE0 6A30 BRT145  GOTO   BRT190
1629
1630 0DCE4      BRT150                           *** ARG>=1
1631 0DCE4 8AC          ?A#0   A
1632 0DCE7 90           GOYES  BRT153
1633 0DCE9 71ED         GOSUB  MAKE1
1634 0DCED 4B0          GOC    BRT154          If 1, go BRT154
1635
1636 0DCF0      BRT153                           *** ARG>1
1637 0DCF0 866          ?ST=0  sATAN
1638 0DCF3 3E           GOYES  BRT144          If ATAN, go BRT144
1639 0DCF5 640F         GOTO   OOPS5
1640
1641 0DCF9      BRT154                           *** ARG=1
1642 0DCF9 876          ?ST=1  sATAN
1643 0DCFC 11           GOYES  BRT175          If ASIN or ACOS, go BRT175
1644 0DCFE 7F9D BRT170  GOSUB  PI/4            No, create pi/4
1645 0DD02 AF5          B=C    W
1646 0DD05 AF0          A=0    W
1647 0DD08 CC           A=A-1  A
1648 0DD0A 4C0          GOC    BRT176          BET
1649 0DD0D 7C7E BRT175  GOSUB  FLIP8           Toggle sCOMP
1650 0DD11 AF0          A=0    W
1651 0DD14 AF1          B=0    W
1652 0DD17 6990 BRT176  GOTO   BRT400
1653
1654 0DD1B 8A8  BRT190  ?A=0   A              Z=1?  **BUGFIX 11/22/82
1655 0DD1E 0E           GOYES  BRT170
1656 0DD20 7AAD         GOSUB  MAKE1           P: 14
1657 0DD24 AF7          D=C    W              D: [0100000000000000]
1658 0DD27 A82          C=0    P              C: [0000000000000000]
1659 0DD2A D6           C=A    A              C: J=I-1=C[S]=0, COUNT=EXP<0
1660 0DD2C E6   BRT191  C=C+1  A              Inc count.  zero?
1661 0DD2E 4F0          GOC    BRT195          Yes, go BRT195
1662 0DD31 B46          C=C+1  S              No, J=J+1
1663 0DD34 0D           P=P-1                  Dec position
1664 0DD36 887          ?P#    7               Done?
1665 0DD39 3F           GOYES  BRT191          No
1666 0DD3B 5BD          GONC   BRT176          BET - tiny Arg (<10^-7)
1667
1668 0DD3E AFF  BRT195  CDEX   W              D: J=I-1;  PQUOT=0
1669 0DD41 4A1          GOC    BRT340          BET. C:X=1,  B:Y'=MANT
1670
1671 0DD44 B07  BRT320  D=D+1  P              PQUOT+1
1672 0DD47 ACB          C=D    S              C: I-1
1673 0DD4A BF4  BRT330  ASR    W              FORM (10^-2I)*Y
1674 0DD4D BF4          ASR    W
1675 0DD50 A4E          C=C-1  S
1676 0DD53 56F          GONC   BRT330
1677 0DD56 AC2          C=0    S
1678 0DD59 A72          C=C+A  W              C:X=X+(10^-2I)*Y
```

```
1679 0DD5C AF4   BRT340 A=B    W          Y=Y' (save current value of Y)
1680 0DD5F B71          B=B-C  W          B:Y'=Y-X.  Y'<0 ?
1681 0DD62 51E          GONC   BRT320     Not yet, go compute new X.
1682 0DD65 AF8          B=A    W          Yes. too far, Y'=Y (Restore)
1683 0DD68 B47          D=D+1  S          Bump I-1 (smaller rotate angle)
1684 0DD6B BF1          BSL    W          Y'=10Y'
1685 0DD6E 0D           P=P-1             Dec PQUOT position
1686 0DD70 887          ?P#    7          Done?
1687 0DD73 9E           GOYES  BRT340     NO
1688
1689 0DD75 AFF          CDEX   W
1690 0DD78 AFD          BCEX   W
1691 0DD7B AF4          A=B    W          A:PQUOT
1692 0DD7E 8E00         GOSUBL =DIV120    B:Y/X
          00
1693 0DD84 28           P=     8          P:NIB PTR = TABLE INDEX
1694
1695 0DD86 791D  BRT350 GOSUB  GETCON     FETCH ATAN(10^-I)
1696 0DD8A 6600         GOTO   BRT370
1697 0DD8E A71   BRT360 B=B+C  W          B:ARG+ATAN(10^-I)
1698 0DD91 A0C   BRT370 A=A-1  P          Dec PQUOT.  Borrow?
1699 0DD94 59F          GONC   BRT360     No
1700 0DD97 A80          A=0    P
1701 0DD9A BF5          BSR    W
1702 0DD9D 958          ?A=0   M          PQUOT=0?
1703 0DDA0 A0           GOYES  BRT390     YES
1704 0DDA2 0C           P=P+1             Bump PTR
1705 0DDA4 61EF         GOTO   BRT350
1706
1707 0DDA8 0C    BRT380 P=P+1
1708 0DDAA CC    BRT390 A=A-1  A          Establish EXP
1709 0DDAC 88E          ?P#    14
1710 0DDAF 9F           GOYES  BRT380
1711
1712 0DDB1 AC0   BRT400 A=0    S
1713 0DDB4 8E00         GOSUBL =SHF10     Shift & Normalize
          00
1714 0DDBA 868          ?ST=0  sCOMP      Need PI/2 - Q ?
1715 0DDBD F0           GOYES  BRT410     No, go BRT410
1716 0DDBF BCC          A=-A-1 S
1717 0DDC2 71BD         GOSUB  PI/2
1718 0DDC6 8E00         GOSUBL =AD15s     PI/2 - Q. Round mode not
          00
1719                                          important--Case by Case.
1720
1721 0DDCC 86B   BRT410 ?ST=0  s+PI/2     Need add PI/2 ?
1722 0DDCF C0           GOYES  BRT420     No, go BRT420
1723 0DDD1 72AD         GOSUB  PI/2       Yes, fetch PI/2
1724 0DDD5 8E00         GOSUBL =AD15s     Round mode not important.
          00
1725                                          Case by Case.
1726
1727
1728            *----NEED: sRAD,sIX,sSGN, A/B=Result----*
1729 0DDDB 879   BRT420 ?ST=1  =sRAD       RAD Mode?
```

```
1730 ODDDE A1          GOYES  BRT430        Yes, go BRT430
1731 ODDE0 27          P=     7             No, convert to DEG.
1732 ODDE2 7DBC        GOSUB  GETCON        Fetch PI/180
1733 ODDE6 AF7         D=C    W
1734 ODDE9 D2          C=0    A
1735 ODDEB CE          C=C-1  A
1736 ODDED CE          C=C-1  A
1737 ODDEF AFD         BCEX   W
1738 ODDF2 8E00        GOSUBL =DIV100       Q*180/PI
           00
1739
1740 ODDF8 7A00  BRT430 GOSUB  FUDGE        Make perfect if exact
1741 ODDFC 86A         ?ST=0  sSGN          Negate?
1742 ODDFF 50          GOYES  BRT440        No.
1743 ODE01 BCC         A=-A-1 S             Yes.
1744 ODE04 03    BRT440 RTNCC
1745
1746 ODE06 8E53  FUDGE GOSUBL SETSB         Ensure SB=1
           8F
1747 ODE0C 877         ?ST=1  sIX           Inexact?
1748 ODE0F 00          RTNYES               Yes, done.
1749
1750 ODE11 822         SB=0                 Exact: Make the 15-dig
1751 ODE14 A35         B=B+B  X             result perfect (round
1752 ODE17 AB1         B=0    X             to 12-digits), set SB=0.
1753 ODE1A 500         RTNNC
1754 ODE1D BBD         B=-B-1 X             B[X]: 999
1755 ODE20 2E          P=     14
1756 ODE22 B15         B=B+1  WP            Round up
1757 ODE25 500         RTNNC
1758 ODE28 E4          A=A+1  A             Bump EXP
1759 ODE2A B05         B=B+1  P             MANT=1.000...
1760 ODE2D 03          RTNCC
1761
1762 ODE2F             END
```

```
 1/X15   Ext                    -   992  1087  1626
 ACNOTO  Abs   54545 #0D511  -   231   225
=ACOS12  Abs   56275 #0DBD3  -  1527
=ACOS15  Abs   56279 #0DBD7  -  1528
 AD15s   Ext                    -  1718  1724
 ADDF    Ext                    -   683
 ADDONE  Ext                    -  1084  1610
 AEQC    Abs   54539 #0D50B  -   229   192   241
 AFIN    Abs   55040 #0D700  -   689   181   324   392   451   644   658   779
                                 1536
 AGTC    Abs   54513 #0D4F1  -   218   195
 AISO    Abs   54534 #0D506  -   227   223
 ALTC    Abs   54449 #0D4B1  -   196   217
=ARG12   Abs   54907 #0D67B  -   638
=ARG15   Abs   54911 #0D67F  -   639
 ARGEX   Abs   55034 #0D6FA  -   683   679
=ARGF    Abs   54948 #0D6A4  -   652   659
=ASIN12  Abs   56264 #0DBC8  -  1523
=ASIN15  Abs   56268 #0DBCC  -  1524
=ATAN12  Abs   56250 #0DBBA  -  1518
=ATAN15  Abs   56254 #0DBBE  -  1519   662
=BIASA+  Abs   54573 #0D52D  -   276   178
=BIASA-  Abs   54573 #0D52D  -   275   200
=BIASC+  Abs   54592 #0D540  -   284   179
=BIASC-  Abs   54592 #0D540  -   283   201
 BLDNAN  Abs   55119 #0D74F  -   786
 BRT10   Abs   56282 #0DBDA  -  1529  1525
 BRT100  Abs   56414 #0DC5E  -  1583  1575
 BRT110  Abs   56422 #0DC66  -  1588  1559  1563  1571  1582  1584
 BRT130  Abs   56444 #0DC7C  -  1598  1590  1593
 BRT135  Abs   56453 #0DC85  -  1601  1599
 BRT140  Abs   56460 #0DC8C  -  1605
 BRT144  Abs   56534 #0DCD6  -  1626  1638
 BRT145  Abs   56544 #0DCE0  -  1628  1606  1625
 BRT150  Abs   56548 #0DCE4  -  1630  1603
 BRT153  Abs   56560 #0DCF0  -  1636  1632
 BRT154  Abs   56569 #0DCF9  -  1641  1634
 BRT170  Abs   56574 #0DCFE  -  1644  1655
 BRT175  Abs   56589 #0DD0D  -  1649  1643
 BRT176  Abs   56599 #0DD17  -  1652  1648  1666
 BRT190  Abs   56603 #0DD1B  -  1654  1628
 BRT191  Abs   56620 #0DD2C  -  1660  1665
 BRT195  Abs   56638 #0DD3E  -  1668  1661
 BRT20   Abs   56285 #0DBDD  -  1532  1521
=BRT30   Abs   56291 #0DBE3  -  1534
 BRT320  Abs   56644 #0DD44  -  1671  1681
 BRT330  Abs   56650 #0DD4A  -  1673  1676
 BRT340  Abs   56668 #0DD5C  -  1679  1669  1687
 BRT350  Abs   56710 #0DD86  -  1695  1705
 BRT360  Abs   56718 #0DD8E  -  1697  1699
 BRT370  Abs   56721 #0DD91  -  1698  1696
 BRT380  Abs   56744 #0DDA8  -  1707  1710
 BRT390  Abs   56746 #0DDAA  -  1708  1703
 BRT40   Abs   56318 #0DBFE  -  1545  1541
 BRT400  Abs   56753 #0DDB1  -  1712  1600  1652
```

```
 BRT410  Abs   56780 #0DDCC -  1721  1715
 BRT420  Abs   56795 #0DDDB -  1729  1551  1722
 BRT430  Abs   56824 #0DDF8 -  1740  1730
 BRT440  Abs   56836 #0DE04 -  1744  1742
 BRT60   Abs   56337 #0DC11 -  1551  1549
 BRT80   Abs   56366 #0DC2E -  1564  1561
=BRTF    Abs   56341 #0DC15 -  1554  1537
 BRTNEX  Abs   56419 #0DC63 -  1585  1565  1568  1572  1577  1579  1581
 BRTRAD  Abs   56389 #0DC45 -  1574  1557
 C+1RTN  Abs   54721 #0D5C1 -   407   399
 CCEXIT  Abs   54650 #0D57A -   336   340
=CLASSA  Abs   54672 #0D590 -   390
=COS12   Abs   55073 #0D721 -   769
=COS15   Abs   55077 #0D725 -   770
 CTRICK  Abs   54488 #0D4D8 -   208   207
 DBL3    Abs   56043 #0DAEB -  1263  1261
 DBL4    Abs   56055 #0DAF7 -  1267  1259
 DBL45   Abs   56105 #0DB29 -  1312  1303
=DBLPI4  Abs   56060 #0DAFC -  1301   848   874   909
=DBLSUB  Abs   56029 #0DADD -  1258   860   886   910   914
 DIV100  Ext                -  1738
 DIV120  Ext                -  1692
 DIVFCD  Ext                -  1064
 DIVYX   Abs   54967 #0D6B7 -   660   649   653   655
 DV2-15  Ext                -   660
 DZINF   Ext                -   467
 DZP     Ext                -  1072
 EGRESS  Abs   56098 #0DB22 -  1309  1316
=EX12    Abs   54726 #0D5C6 -   449
=EX15M   Abs   54730 #0D5CA -   450
=EX15S   Abs   54734 #0D5CE -   451
 EXAB1   Ext                -  1611
=EXF     Abs   54751 #0D5DF -   459   452
 EXIT02  Abs   54777 #0D5F9 -   468   454
 EXIT03  Abs   54388 #0D474 -   119   105   111
 EXLOOP  Abs   54797 #0D60D -   478   474   481
 FINA    Abs   54493 #0D4DD -   211   182
 FINAC   Abs   54519 #0D4F7 -   221   213
 FINITA  Ext                -   689
 FINITC  Ext                -   211   337
=FLIP10  Abs   56220 #0DB9C -  1453   891  1080
=FLIP11  Abs   56235 #0DBAB -  1460   888
=FLIP8   Abs   56205 #0DB8D -  1446   889   915  1627  1649
 FUDGE   Abs   56838 #0DE06 -  1746  1089  1740
 FX      Abs   54652 #0D57C -   337   325
 FY      Abs   54645 #0D575 -   334   327
=GETCON  Abs   55971 #0DAA3 -  1176   970  1003  1695  1732
=GETVAL  Abs   55986 #0DAB2 -  1180
 INVNaN  Ext                -   786
=IVARG   Abs   55113 #0D749 -   784   650  1543
 IVP     Ext                -   116
 IX      Abs   54661 #0D585 -   339   329
 LOGIC   Abs   54451 #0D4B3 -   197   186   190   215   219   230
=MAKE1   Abs   56014 #0DACE -  1218   390  1560  1580  1633  1656
 MAKEPI  Abs   55025 #0D6F1 -   680   673
```

```
 MP2-15  Ext                  -  1083
=MSN12   Abs   54611 #0D553  -   323
=MSN15   Abs   54615 #0D557  -   324   640
 MULTF   Ext                  -   973  1614
 NONFIN  Abs   54702 #0D5AE  -   400   393
 NORM    Abs   55418 #0D87A  -   929   925   933
 OOPS3   Abs   54367 #0D45F  -   112
 OOPS4   Abs   55113 #0D749  -   783
 OOPS5   Abs   56314 #0DBFA  -  1543  1639
=ORGSB   Abs   54875 #0D65B  -   584   663
=ORSB    Abs   54844 #0D63C  -   537   535
=ORXM    Abs   54835 #0D633  -   534
=PI/2    Abs   56183 #0DB77  -  1420  1545  1717  1723
=PI/2D   Abs   56186 #0DB7A  -  1421   680
=PI/4    Abs   55969 #0DAA1  -  1136  1304  1421  1644
 RCCD1   Ext                  -  1398
=RCCD1X  Abs   56179 #0DB73  -  1398  1082  1613
 RCCD2   Ext                  -  1617
 RED360  Abs   55259 #0D7DB  -   859   865
 RED90   Abs   55309 #0D80D  -   886   877   890   892
 REDUCE  Abs   55234 #0D7C2  -   846   827   834
=SAVESB  Abs   54894 #0D66E  -   592   590
=SAVEXM  Abs   54883 #0D663  -   588   532
=SAVGSB  Abs   54862 #0D64E  -   578   661
 SBOCC   Abs   54812 #0D61C  -   485
=SETSB   Abs   54849 #0D641  -   539   586  1746
 SETXM   Abs   54860 #0D64C  -   544   536
 SHF10   Ext                  -  1713
=SHFLAC  Abs   56134 #0DB46  -  1360   862   901   931
=SHFRAC  Abs   56145 #0DB51  -  1365   838   879
=SHFRBD  Abs   56159 #0DB5F  -  1388   852   878
 SIGNA   Abs   54444 #0D4AC  -   194   188   226   232   234   238
 SIGNC   Abs   54508 #0D4EC  -   216   228   236   240
=SIN12   Abs   55062 #0D716  -   765
=SIN15   Abs   55066 #0D71A  -   766
 SPLITA  Ext                  -   690
 SPLTA   Abs   55046 #0D706  -   690   449   765   769   774  1518  1523  1527
 SPLTAC  Ext                  -   691
 SPLTB   Abs   55052 #0D70C  -   691   174   323   638
 SQR15   Ext                  -  1086
 SQR15M  Ext                  -   533
 SQR17   Ext                  -  1616
=SQRSAV  Abs   54825 #0D629  -   532
 STAB1   Ext                  -   693
=STAB1X  Abs   55058 #0D712  -   693  1081  1608
 STAB2   Ext                  -  1609
 SUBONE  Ext                  -  1612
=SWAPXY  Abs   55819 #0DA0B  -  1103   341   645  1546
=TAN12   Abs   55087 #0D72F  -   774
=TAN15   Abs   55091 #0D733  -   775
 TOG10   Abs   56230 #0DBA6  -  1457  1454
 TOG11   Abs   56245 #0DBB5  -  1464  1461
 TOG8    Abs   56215 #0DB97  -  1450  1447
=TRC90   Abs   55825 #0DA11  -  1125  1179
 TRG10   Abs   55080 #0D728  -   771   767
```

TRG100	Abs	55145	#0D769	-	795	793					
TRG125	Abs	55162	#0D77A	-	802	800					
TRG130	Abs	55178	#0D78A	-	808	803	806				
TRG135	Abs	55196	#0D79C	-	816	814					
TRG155	Abs	55262	#0D7DE	-	860	861					
TRG160	Abs	55285	#0D7F5	-	873						
TRG180	Abs	55335	#0D827	-	895	887					
TRG181	Abs	55338	#0D82A	-	896	831	847				
TRG260	Abs	55361	#0D841	-	904	899					
TRG270	Abs	55364	#0D844	-	906	897	903				
TRG275	Abs	55397	#0D865	-	920	911					
TRG279	Abs	55433	#0D889	-	940	907					
TRG280	Abs	55436	#0D88C	-	941	927	930				
TRG290	Abs	55485	#0D8BD	-	966	944	948	955	958		
TRG300	Abs	55509	#0D8D5	-	978	968					
TRG310	Abs	55524	#0D8E4	-	985	980					
TRG314	Abs	55547	#0D8FB	-	993	991					
TRG318	Abs	55551	#0D8FF	-	995	989					
TRG330	Abs	55563	#0D90B	-	1002	983					
TRG340	Abs	55566	#0D90E	-	1003	1013					
TRG350	Abs	55630	#0D94E	-	1031	1034					
TRG360	Abs	55636	#0D954	-	1033	1042					
TRG370	Abs	55654	#0D966	-	1039	1024	1050				
TRG400	Abs	55690	#0D98A	-	1055	1045					
TRG420	Abs	55713	#0D9A1	-	1064	1061					
TRG430	Abs	55719	#0D9A7	-	1065	993					
TRG440	Abs	55743	#0D9BF	-	1075	1066					
TRG450	Abs	55760	#0D9D0	-	1081	1077	1079				
TRG50	Abs	55097	#0D739	-	778	772					
TRG500	Abs	55792	#0D9F0	-	1089	1068					
TRG510	Abs	55796	#0D9F4	-	1090						
TRG515	Abs	55801	#0D9F9	-	1092	1073					
TRG520	Abs	55806	#0D9FE	-	1094	1098					
TRG530	Abs	55811	#0DA03	-	1096	1091					
TRG800	Abs	55576	#0D918	-	1006	1004					
TRG810	Abs	55573	#0D915	-	1005	1007					
TRGEX	Abs	55482	#0D8BA	-	959	946					
TRGF	Abs	55125	#0D755	-	788	780					
TRGTST	Abs	55477	#0D8B5	-	957	953					
=TST12A	Abs	54390	#0D476	-	174	101					
=TST15	Abs	54394	#0D47A	-	175						
=TWO*	Abs	56120	#0DB38	-	1337	849	850	851	875		
TWONaN	Ext			-	332						
X/Y15	Ext			-	1618						
XFINIT	Abs	54960	#0D6B0	-	658	646					
XMOSBO	Abs	54817	#0D621	-	527	450	639	778			
XNONO	Abs	54779	#0D5FB	-	470	462					
XYEX	Ext			-	1103						
eEXPO	Ext			-	466						
eIVARG	Ext			-	785						
eTNINF	Ext			-	1071						
eUNORC	Ext			-	114						
s+PI/2	Abs	11	#0000B	-	63	1535	1595	1721			
sATAN	Abs	6	#00006	-	62	1520	1529	1540	1564	1605	1637	1642
sCOMP	Abs	8	#00008	-	61	1519	1524	1528	1574	1592	1594	1714

sINVRT	Abs	8	#00008	-	56	766	770	775	792	805	954	990
					1060	1446	1448	1450				
sINX	Ext			-	578	581	584					
sIX	Abs	7	#00007	-	55	117	537	592	595	798	801	959
					1533	1550	1585	1747				
=sRAD	Abs	9	#00009	-	53	672	826	876	896	943	967	1302
					1548	1556	1729					
sSGN	Abs	10	#0000A	-	58	790	807	1096	1453	1455	1457	1534
					1591	1596	1741					
sSGNT	Abs	11	#0000B	-	59	791	804	1092	1460	1462	1464	
sTAN	Abs	6	#00006	-	57	771	776	952	1065	1090		
sXM	Abs	9	#00009	-	54	534	588	591				
uRESXT	Ext			-	118							
=uTEST	Abs	54325	#0D435	-	98							

Input Parameters

  Source file name is SM&MTH::MS

  Listing file name is SM/MTH:TI:ML::-1

  Object file name is SM%MTH:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                      TITLE  Statistics <831216.1457>
 2 ODE2F                ABS    #ODE2F
 3                *
 4                *  PPPP   M   M   &      SSS   TTTTT    A
 5                *  P   P  MM MM  & &    S   S    T     A A
 6                *  P   P  M M M  & &    S        T    A   A
 7                *  PPPP   M M M   &      SSS     T    A   A
 8                *  P      M   M  & & &      S    T    AAAAA
 9                *  P      M   M  &  &   S   S    T    A   A
10                *  P      M   M   && &   SSS     T    A   A
11                *
12                ***************************************************************
13                ***************************************************************
14                **
15                ** Name:    DROP    -  Removes data point from statistics
16                **
17                ** Category:   STEXEC
18                **
19                ** Purpose:
20                **      Drops a data point from the currently specified
21                **      statistical array.  See the documentation for ADD for
22                **      further information.
23                **
24                ***************************************************************
25                ***************************************************************
26 ODE2F 0000          REL(5) =DROPDC
         0
27 ODE34 0000          REL(5) =DROPP
         0
28 ODE39 74F3 =DROP    GOSUB  TEXPEX          build expression stack
29 ODE3D 850           ST=1   =sSTAT          set DROP status
30 ODE40 6830          GOTO   DENTRY
```

```
                        EJECT
                *****************************************************************
                *****************************************************************
                **
                ** Name:    CLSTAT  -  Sets current stat array to zero
                **
                ** Category:   STEXEC
                **
                ** Purpose:
                **      Clears the current statistical array (sets all elements
                **      to zero).
                **
                ** Entry:
                **      current statistical array specified
                **
                ** Exit:
                **      current statistical array cleared
                **
                ** Calls:      GETMSG,GETSA
                **             exits through NXTSTM
                **
                ** Uses.......
                **  Inclusive: A,B,C(6-0),D(A),P,D0,R2,SB,XM,
                **             statistical array,  unless fatal error
                **
                ** Stk lvls:   3
                **
                ** NOTE:
                **      The current statistical array's dope vector is not
                **      altered, so the current regression is unchanged.
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **   --------   ----------   ---------------------------------
                **   06/10/82      PM          Documented routine
                **   01/06/83       "          Revised documentation
                **
                *****************************************************************
                *****************************************************************
ODE44 0000          REL(5) =STOPDC
      0
ODE49 0000          REL(5) =RTNCC
      0
ODE4E 7FF6 =CLSTAT  GOSUB  GETSA            D0 points to N,
ODE52 AF0           A=0    W                C(A)=length in hex
ODE55 CE    CLSTLP  C=C-1  A
ODE57 4C0           GOC    nxtst1           done?
ODE5A 1507          DAT0=A W
ODE5E 16F           D0=D0+ 16
ODE61 53F           GONC   CLSTLP           B.E.T.
                *****************************************************************
ODE64 62E1  nxtst1  GOTO   nxtstm
                *****************************************************************
                *****************************************************************
```

```
 84                **
 85                ** Name:    ADD     -  Adds data point to statistics
 86                **
 87                ** Category:   STEXEC
 88                **
 89                ** Purpose:
 90                **      Adds a data point into the currently specified
 91                **      statistical array.  The data point may have 0-15
 92                **      specified coordinates (variables).  If less than
 93                **      15 are specified, any missing ones are assumed zero.
 94                **      The accumulated summary statistics  are the number
 95                **      of data points, the totals, and the mean-adjusted
 96                **      sums of squares and cross products.
 97                **      A recursive method is used to update the statistics.
 98                **
 99                ** Entry:
100                **      variables in parsed form
101                **      current statistical array specified
102                **
103                ** Exit:
104                **      current statistical array updated
105                **
106                ** Calls:      ADDONE,AD15S,ARESD1,BFGTAB,BFGTCD,BFPTAB,
107                **             D0=SR0,D1=SR1,D1+21,FBWVAR,GETSA,I/ODAL,MESSG,
108                **             MSARES,NNGTAB,RCCD1,STAB1,STBALL,SPLTAS,
109                **             TEXPEX,XYEX,dv15s,mp15s,rclw1,rclw1+,spltA+,
110                **             stscr,subone,uRESD1,umode+
111                **             exits through NXTSTM
112                **
113                ** Uses.......
114                **  Exclusive of EXPEX-:
115                **       CPU:  A,B,C,D,P,D0,D1,R0-4,SB,XM,sSTAT(s6),sIX(s7),
116                **             s8-11
117                **       RAM:  scratch math stack
118                **             statistical buffer
119                **             statistical array
120                **             S-R0-0,1,2,3, STMTD1, MTHSTK/AVMEME
121                **
122                ** Stk lvls:   max(6,1+EXPEX-) = max(1+EXPEX-,3+uRESD1)
123                **
124                **
125                ** History:
126                **
127                **     Date     Programmer            Modification
128                **    --------   ----------   ---------------------------------
129                **    06/07/82      PM         Documented routine
130                **    12/16/82       "         Added tests for signaling NaNs
131                **    01/31/83       "         Handled arithmetic-created NaNs
132                **    03/04/83       "         Modified ADD algorithm, packed
133                **
134                ********************************************************************
135                ********************************************************************
136 0DE68 0000          REL(5) =ADDDC
           0
137 0DE6D 0000          REL(5) =ADDP
```

```
          0
  138 0DE72 7BB3 =ADD    GOSUB  TEXPEX          build expression stack
  139 0DE76 840          ST=0   =sSTAT          set ADD status
  140 0DE79 74D6  DENTRY GOSUB  GETSA           get SA address, #VARS
  141 0DE7D 75F1         GOSUB  STBALL          allocate and fill buffer, scr RAM
  142 0DE81 78C3         GOSUB  stscr           W1:=old N
  143 0DE85 7F83         GOSUB  umodeH          get user modes, clear SB,XM, sethex
  144             ********************************************************************
  145             *  COMPUTE ADJUSTED SS AND SCP  (have hexmode, P=14)
  146             ********************************************************************
  147 0DE89 72D1  ADJSQ  GOSUB  D1=SR1          point D1 to buffer
  148 0DE8D 7EB1         GOSUB  D0=SR0          C(S):=#VARS, point D0 to N
  149 0DE91 A82          C=0    P               C(14)=#SS-1:=0
  150 0DE94 A4E  ADJSQ2  C=C-1  S               no more variables?
  151 0DE97 495          GOC    ADJSQ4
  152 0DE9A 10C          R4=C
  153 0DE9D 16F  ADJSQ3  D0=D0+ 16              point D0 to next total
  154 0DEA0 A0E          C=C-1  P
  155 0DEA3 59F          GONC   ADJSQ3
  156 0DEA6 1537         A=DAT1 W               get VJ (12-digit mantissa)
  157 0DEAA 7192         GOSUB  SPLTAS          split, normalize, test for signan
  158 0DEAE 7F93         GOSUB  rclw1+          get N (15-digit), save D0, setdec
  159 0DEB2 7DF2         GOSUB  mp15s           N*VJ
  160 0DEB6 4B1          GOC    ADJSQN          jump if Inf*0 NaN created
  161 0DEB9 11B          C=R3
  162 0DEBC 134          D0=C                   restore D0
  163 0DEBF 7000         GOSUB  =STAB1X         store product in R0/R1
  164 0DEC3 7472         GOSUB  spltA+          get, split, norm., test total TJ
  165 0DEC7 7000         GOSUB  =RCCD1X         recall product into C/D
  166 0DECB BCC          A=-A-1 S
  167 0DECE 7BC2         GOSUB  AD15S           Inf-Inf?
  168 0DED2 7A92  ADJSQN GOSUB  BFPTAB          FJ:=N*VJ-TJ (15-digit), P:=14
  169 0DED6 76F2         GOSUB  RPTNaN          Report any invalid NaN, P:=14
  170 0DEDA 7091         GOSUB  D1+21           point D1 to next variable
  171 0DEDE 04           SETHEX
  172 0DEE0 11C          C=R4
  173 0DEE3 B06          C=C+1  P               #SS:=#SS+1  (P=14 here)
  174 0DEE6 5DA          GONC   ADJSQ2          B.E.T.
  175             ********************************************************************
  176 0DEE9 7B33  ABORTS GOSUB  GETMSG          get error message
  177 0DEED 6A32         GOTO   mferr
  178             ********************************************************************
  179 0DEF1 7263  ADJSQ4 GOSUB  rclw1           recall N (15 digit), setdec
  180 0DEF5 7192         GOSUB  addsub          N+1 or N-1
  181 0DEF9 7A53         GOSUB  rclw1           recall N
  182 0DEFD 860          ?ST=0  =sSTAT          ADD?
  183 0DF00 01           GOYES  SQADD2
  184 0DF02 7606         GOSUB  POSTAB          DROP: error if N<=0
  185 0DF06 42E          GOC    ABORTS
  186 0DF09 7000         GOSUB  =SWAPXY               exchange N and N-1
  187 0DF0D BCE          C=-C-1 S                     negate N
  188 0DF10 7206  SQADD2 GOSUB  NNGTAB
  189 0DF14 44D          GOC    ABORTS          error: N<0 (ADD) or 0<N<1 (DROP)
  190 0DF17 7892         GOSUB  mp15s           F0:=N*(N+1) or -N*(N-1)
  191                                              (Inf*0 not possible)
```

```
192 0DF1B 7E23          GOSUB  stscr          W1:=F0
193 0DF1F 04            SETHEX
194 0DF21 7A21          GOSUB  D0=SR0         point D0 to N
195 0DF25 10C           R4=C                  R4(S)=#VARS
196 0DF28 7331          GOSUB  D1=SR1         point C(A),D1 to F1
197 0DF2C AC1           B=0    S
198 0DF2F 97D           ?B#0   W              F0#0?
199 0DF32 92            GOYES  ADJSQ6
200                 ****************************************************************
201                 *  SET SS AND SCP TO ZERO
202                 ****************************************************************
203 0DF34 AF0           A=0    W              need hex mode
204 0DF37 16F           D0=D0+ 16             D0 points to total
205 0DF3A 11C           C=R4                  C(S):=#VARS
206 0DF3D D2            C=0    A              C(A)=(#SS to be set to zero)-1
207 0DF3F A4E   ZEROS1  C=C-1  S              done?
208 0DF42 4E1           GOC    ATOTO
209 0DF45 D5            B=C    A              B(A):=#SS-1
210 0DF47 16F           D0=D0+ 16
211 0DF4A 1507  ZEROS2  DAT0=A W              SS:=0
212 0DF4E 16F           D0=D0+ 16
213 0DF51 CD            B=B-1  A
214 0DF53 56F           GONC   ZEROS2
215 0DF56 E6            C=C+1  A              #SS:=#SS+1
216 0DF58 56E           GONC   ZEROS1         B.E.T.
217                 ****************************************************************
218                 *  COMPUTE NONZERO ADJUSTED SS AND SCP
219                 ****************************************************************
220 0DF5B 114   ADJSQ6  A=R4                  need hex mode, XM=0
221 0DF5E A4C           A=A-1  S              no more variables?
222 0DF61 4B7   ATOTO   GOC    ADJTOT
223 0DF64 104           R4=A
224 0DF67 1F00          D1=(5) =S-R0-1        point D1 to F1
          000
225 0DF6E 143           A=DAT1 A
226 0DF71 131           D1=A
227 0DF74 16F           D0=D0+ 16             skip total
228 0DF77 16F   ADJSQ7  D0=D0+ 16             point D0 to next SCP or SS
229 0DF7A 133           AD1EX                 see if is a SS
230 0DF7D 131           D1=A
231 0DF80 06            RSTK=C                save C(A) in RSTK
232 0DF82 8A2           ?A=C   A              SS?
233 0DF85 52            GOYES  ADJSQ8
234 0DF87 75C1          GOSUB  BFGTAB         no: get FI/F0, I<J
235 0DF8B 07            C=RSTK                get FJ
236 0DF8D 137           CD1EX
237 0DF90 06            RSTK=C
238 0DF92 7AC1          GOSUB  BFGTCD
239 0DF96 7E32          GOSUB  MSARES         Inf*0,Inf-Inf NaNs handled
240 0DF9A 1547          DAT0=C W              SCP:=[SCP+FI/F0*FJ]
241 0DF9E 07            C=RSTK                restore D1, C(A)
242 0DFA0 137           CD1EX
243 0DFA3 77C0          GOSUB  D1+21          point D1 to next term
244 0DFA7 5FC           GONC   ADJSQ7         B.E.T.
245 0DFAA 73A2  ADJSQ8  GOSUB  rclw1+         SS: get F0#0, setdec
```

```
 246 0DFAE 11B          C=R3                    restore D0
 247 0DFB1 134          D0=C
 248 0DFB4 7000         GOSUB  =SWAPXY          move F0 to C/D
 249 0DFB8 7491         GOSUB  BFGTAB           get FJ
 250 0DFBC 7245         GOSUB  dv15s            FJ/F0.
 251                                            Inf/Inf? (0/0,DVZ not possible)
 252 0DFC0 7C91         GOSUB  BFGTCD
 253 0DFC4 78A1         GOSUB  BFPTAB
 254 0DFC8 7C02         GOSUB  MSARES           handles Inf/Inf.
 255                                             (Inf*0,Inf-Inf not possible)
 256 0DFCC 1547         DAT0=C W                SS:=[SS+FJ*FJ/F0], FJ:=FJ/F0
 257 0DFD0 7A90         GOSUB  D1+21            point to next FJ
 258 0DFD4 07           C=RSTK                  store in C(A)
 259 0DFD6 137          CD1EX
 260 0DFD9 618F         GOTO   ADJSQ6
 261 0DFDD 20   ADJTOT  P=     0
 262 0DFDF 75D0         GOSUB  FBWVAR           fill buffer with vars, setdec
 263 0DFE3 73A1         GOSUB  addsub           N+1 or N-1
 264 0DFE7 7302         GOSUB  uRESD1
 265 0DFEB 1547         DAT0=C W                N:=[N+1] or :=[N-1]
 266            ********************************************************************
 267            *  COMPUTE NEW TOTALS
 268            ********************************************************************
 269 0DFEF 11C          C=R4
 270 0DFF2 A82          C=0    P                C(14)=#SS-1:=0
 271 0DFF5 A4E  ATOT1   C=C-1  S                no more variables?
 272 0DFF8 414          GOC    ADDEND
 273 0DFFB 10C          R4=C
 274 0DFFE 16F  ATOT2   D0=D0+ 16               point D0 to next total TJ
 275 0E001 A0E          C=C-1  P
 276 0E004 59F          GONC   ATOT2
 277 0E007 7031         GOSUB  spltA+           get, split, normalize, test TJ
 278 0E00B 8E00         GOSUBL =STAB1           save TJ in R0/R1
           00
 279 0E011 1537         A=DAT1 W                get variable VJ (12-digit)
 280 0E015 7621         GOSUB  SPLTAS           split normalize, test VJ
 281 0E019 7000         GOSUB  =RCCD1X          recall TJ
 282 0E01D 7D40         GOSUB  D1+21            point D1 to next variable
 283 0E021 860          ?ST=0  =sSTAT
 284 0E024 50           GOYES  ATOT3
 285 0E026 BCC          A=-A-1 S
 286 0E029 7DB1 ATOT3   GOSUB  ARESD1           any Inf-Inf NaN is recorded
 287 0E02D 1547         DAT0=C W                TJ:=[TJ+VJ] or :=[TJ-VJ]
 288 0E031 11C          C=R4
 289 0E034 B06          C=C+1  P                #SS:=#SS+1
 290 0E037 5DB          GONC   ATOT1            B.E.T. = branch every time
 291 0E03A 20   ADDEND  P=     0                done: deallocate stat buffer
 292 0E03C 3200         LC(3)  =bSTAT
           0
 293 0E041 8E00         GOSUBL =I/ODAL
           00
 294 0E047 04   nxtstm  SETHEX
 295 0E049 8C00         GOLONG =NXTSTM          --- exit ---
           00
```

```
296                     EJECT
297                 ********************************************************************
298                 * D0=SR0      S.L.=0     Uses: C,D0
299                 ********************************************************************
300 0E04F 1B00   D0=SR0 D0=(5) =S-R0-0
          000
301 0E056 1567          C=DAT0 W
302 0E05A 134           D0=C
303 0E05D 01            RTN
304                 ********************************************************************
305                 * D1=SR1      S.L.=0     Uses: C(A),D1
306                 ********************************************************************
307 0E05F 1F00   D1=SR1 D1=(5) =S-R0-1
          000
308 0E066 147           C=DAT1 A
309 0E069 135           D1=C
310 0E06C 01            RTN
311                 ********************************************************************
312 0E06E 174    D1+21  D1=D1+ 5
313 0E071 17F           D1=D1+ 16
314 0E074 03            RTNCC
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:    STBALL  -  Allocate and Fill Statistical Buffer
                **
                ** Category:   CONFIG
                **
                ** Purpose:
                **      Allocates the statistics scratch buffer and fills it
                **      with the value of the variables specified by an ADD or
                **      DROP statement.  List length > # variables is
                **      interpreted as a fatal error.
                **
                ** Entry:
                **      P=0
                **      A(S) -- # variables
                **      A(A) -- statistical array starting address
                **      D1 ---- points to top of math stack
                **
                ** Exit:
                **      MTHSTK/AVMEME - point to top of math stack
                **      S-R0-0 (5) ---- N address
                **      S-R0-1 (5) ---- i/o buffer starting address
                **      S-R0-2 (5) ---- junk???
                **      S-R0-3 (1) ---- # variables
                **      also FBWVAR outputs
                **      -- unless fatal error --
                **
                ** Calls:       ave=d1,D0=SR0,D1=SR1,D1+21,FBTEST,I/OALL,
                **              SPLTAX,uRESD1
                **              falls into FBWVAR
                **
                ** Uses.......
                **  Inclusive: CPU: A,B,C,D,P,R3,R4,D0,D1,SB,XM,sIX(s7),s8-11
                **             RAM: MTHSTK/AVMEME,STMTD1,S-R0-0,1,2,3
                **                  statistical buffer
                **
                ** Stk lvls:   4 = max(3,1+uRESD1)
                **
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ------------------------------------
                **  06/10/82       PM         Documented routine
                **  11/03/82        "         Updated MTHSTK/AVMEME
                **  01/14/83        "         Fatal error if list length > #vars
                **  01/27/83        "         S-R0-2 not set to 00000
                **
                ********************************************************************
                ********************************************************************
OE076 8E00  STBALL GOSUBL =ave=d1        update MTHSTK/AVMEME
      00
OE07C 1F00         D1=(5) =S-R0-0
```

```
            000
   369 0E083 1517          DAT1=A W            S-R0-0,3:=N address,#VARS
   370 0E087 D0            A=0    A
   371 0E089 810           ASLC                A(A)=#VARS
   372 0E08C D8            B=A    A
   373 0E08E C5            B=B+B  A
   374 0E090 C5            B=B+B  A
   375 0E092 C8            B=A+B  A
   376 0E094 F0            ASL    A
   377 0E096 C8            B=A+B  A            B(A):=21*(#VARS)
   378 0E098 3200          LC(3)  =bSTAT
            0
   379 0E09D 8E00          GOSUBL =I/OALL      allocate statistics buffer
            00
   380 0E0A3 460           GOC    STBAL1       sufficient memory?
   381 0E0A6 6180          GOTO   mferr        no: "eMEM" fatal error
   382 0E0AA 137   STBAL1 CD1EX                 yes: save buffer address
   383 0E0AD 1F00          D1=(5) =S-R0-1
            000
   384 0E0B4 15D4          DAT1=C 5            S-R0-1:=buffer address
```

```
385                     EJECT
386                 ****************************************************************
387                 ****************************************************************
388                 **
389                 ** Name:    FBWVAR  -  Fills statistical scratch buffer
390                 **
391                 ** Category:   LOCAL
392                 **
393                 ** Purpose:
394                 **      Fills the statistics buffer with values placed on the
395                 **      math stack.  If any expected values are missing, zero
396                 **      is used in their place.  Complex, array, and extra
397                 **      values are interpreted as fatal error conditions.
398                 **
399                 ** Entry:
400                 **      P=0
401                 **      HEXMODE
402                 **
403                 ** Exit:
404                 **      buffer filled with 12-digit forms separated by 5 nibs
405                 **           each.
406                 **      D0 ---- points to N
407                 **      D1 ---- points to buffer (just past buffer header)
408                 **      R4(S) - # variables
409                 **      A/B --- N split and normalized
410                 **      Carry=Clear
411                 **      DECMODE
412                 **      -- unless fatal error --
413                 **
414                 ** Calls:      D0=SR0,D1=SR1,D1+21,SPLTAX,uRESD1
415                 **             falls into SPLTAS
416                 **
417                 ** Uses.......
418                 **  Inclusive: CPU: A,B,C,D,P,R3,R4,D0,D1,SB,XM,sIX(s7),s8-11
419                 **             RAM: statistical buffer,STMTD1
420                 **
421                 ** Stk lvls:   4 = max(2,1+uRESD1)
422                 **
423                 ** History:
424                 **
425                 **    Date     Programmer            Modification
426                 **  --------   ----------   ----------------------------------
427                 **  06/10/82      PM         Documented routine
428                 **  12/15/82       "         Added signaling NaN test
429                 **  01/14/83       "         Fatal error if list length > #vars
430                 **  03/21/83       "         Moved error exit
431                 **  04/26/83       "         Merged subroutine, added 16n test
432                 **
433                 ****************************************************************
434                 ****************************************************************
435 0E0B8 30A    FBWVAR LCHEX  A              store A in B(0)
436 0E0BB D5            B=C    A
437 0E0BD 7E9F          GOSUB  D1=SR1         point D1 to buffer
438 0E0C1 1B00          D0=(5) =S-R0-3        C(S):=#VAR
          000
```

```
    439 0E0C8 1564          C=DAT0 S
    440 0E0CC 1A00          D0=(4) =MTHSTK
              00
    441 0E0D2 146           C=DAT0 A              C(A):=MTHSTK
    442 0E0D5 164           D0=D0+ 5
    443 0E0D8 142           A=DAT0 A              A(A):=FORSTK
    444 0E0DB 906           ?A#C   P              stack length not 16n?
    445 0E0DE 72            GOYES  ivarty
    446 0E0E0 A4E   FBWVR1  C=C-1  S              #VAR:=#VAR-1
    447 0E0E3 4B3           GOC    FBWVR4
    448 0E0E6 8BA           ?A<=C  A              out of data?
    449 0E0E9 22            GOYES  FBWVR2
    450 0E0EB 130           D0=A                  no: get and test next value
    451 0E0EE 181           D0=D0- 2              test for complex header
    452 0E0F1 14A           A=DAT0 B
    453 0E0F4 988           ?A>=B  P
    454 0E0F7 E0            GOYES  ivarty
    455 0E0F9 18D           D0=D0- 14             test for array dope vector
    456 0E0FC 1527          A=DAT0 W
    457 0E100 984           ?A<B   P
    458 0E103 E0            GOYES  FBWVR3         carry set if not
    459 0E105 8CFF  ivarty  GOLONG cnflct         data type error
              70
    460 0E10B 130   FBWVR2  D0=A
    461 0E10E AF0           A=0    W              use 0 for missing data
    462 0E111 1517  FBWVR3  DAT1=A W              put real value into buffer
    463 0E115 755F          GOSUB  D1+21
    464 0E119 132           ADOEX
    465 0E11C 53C           GONC   FBWVR1         B.E.T.
    466 0E11F 8BA   FBWVR4  ?A<=C  A              error if any extra data
    467 0E122 E0            GOYES  FBWVR5
    468 0E124 3100          LC(2)  =eIVSOP        (P=0)
    469 0E128       =MFERR0
    470 0E128 20    mferr   P=     0
    471 0E12A 8C00          GOLONG =MFERR
              00
    472 0E130 7B2F  FBWVR5  GOSUB  D1=SR1         point D1 to buffer
    473 0E134 771F          GOSUB  D0=SR0         point D0 to N
    474 0E138 10C           R4=C                  R4(S):=#VARS
    475 0E13B 1527  spltA+  A=DAT0 W              get N
```

```
476                     EJECT
477                 ****************************************************************
478                 ****************************************************************
479                 **
480                 ** Name:    SPLTAS  -  Split, Test Argument for Signaling NaN
481                 **
482                 ** Category:   MTHUTL
483                 **
484                 ** Purpose:
485                 **      Splits, normalizes, tests argument for signaling NaN.
486                 **      If a signaling NaN, raises IVL flag, consults IVL trap,
487                 **      and prints any warning message or saves any fatal error
488                 **      message.
489                 **
490                 ** Entry:
491                 **      A ---- 12-digit form of argument
492                 **
493                 ** Exit:
494                 **      A/B -- split and normalized argument
495                 **
496                 ** Calls:      SPLTAX,uRESD1
497                 **
498                 ** Uses.......
499                 **  Inclusive: see uRESD1
500                 **
501                 ** Stk lvls:   4 = max(2,1+uRESD1)
502                 **
503                 ** Note:
504                 **      Foreign NaNs are treated as signaling NaNs.
505                 **
506                 ** History:
507                 **
508                 **    Date     Programmer              Modification
509                 **  --------   ----------   ------------------------------------
510                 **  12/21/82      PM         Documented routine
511                 **  01/26/83      PM         Revised documentation
512                 **
513                 ****************************************************************
514                 ****************************************************************
515 0E13F 78E4 =SPLTAS GOSUB  SPLTAX         split, normalize, test
516 0E143 500       RTNNC
517 0E146 74A0      GOSUB  uRESD1               raise IVL flag, consult trap
518 0E14A AFA       A=C    W
519 0E14D 51F       GONC   SPLTAS         B.E.T. with quiet NaN
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:    BFGTAB  -  Statistical buffer I/O routine
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Buffer i/o routine: copies 15-digit buffer value into
                **      registers A/B.
                **
                ** Entry:
                **      D1 -- points to 15-digit buffer value
                **
                ** Exit:
                **      A/B - contains 15-digit buffer value
                **      Carry=Clear
                **
                ** Calls:      nothing
                **
                ** Uses.......
                **  Inclusive: A,B
                **
                ** Stk lvls:   0
                **
                ** NOTE:
                **      See ADD for how this routine may be used.
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ----------------------------------
                **  06/10/82       PM         Documented routine
                **  01/06/83       PM         Reviewed documentation
                **
                ****************************************************************
                ****************************************************************
0E150 1537  BFGTAB A=DAT1 W              sign and mantissa
0E154 17F          D1=D1+ 16
0E157 AF8          B=A    W
0E15A 143          A=DAT1 A              exponent
0E15D 542          GONC   D1-16          B.E.T.
```

```
563                     EJECT
564                 ****************************************************************
565                 ****************************************************************
566                 **
567                 ** Name:    BFGTCD  -  Statistical buffer I/O routine
568                 **
569                 ** Category:   LOCAL
570                 **
571                 ** Purpose:
572                 **      Buffer i/o routine: copies 15-digit buffer value into
573                 **      registers C/D.
574                 **
575                 ** Entry:
576                 **      D1 -- points to 15-digit buffer value
577                 **
578                 ** Exit:
579                 **      C/D - contains 15-digit buffer value
580                 **      Carry=Clear
581                 **
582                 ** Calls:      nothing
583                 **
584                 ** Uses.......
585                 **  Inclusive: C,D
586                 **
587                 ** Stk lvls:   0
588                 **
589                 ** NOTE:
590                 **      Same as BFGTAB, except uses registers C,D.
591                 **
592                 ** History:
593                 **
594                 **    Date      Programmer              Modification
595                 **  --------    ----------   ----------------------------------
596                 **  06/10/82       PM         Documented routine
597                 **  01/06/83        "         Reviewed documentation
598                 **
599                 ****************************************************************
600                 ****************************************************************
601 0E160 1577  BFGTCD C=DAT1 W              sign and mantissa
602 0E164 17F          D1=D1+ 16
603 0E167 AF7          D=C    W
604 0E16A 147          C=DAT1 A              exponent
605 0E16D 541          GONC   D1-16          B.E.T.
```

```
606                     EJECT
607                 ********************************************************************
608                 ********************************************************************
609                 **
610                 ** Name:    BFPTAB  -  Statistical buffer I/O routine
611                 **
612                 ** Category:   LOCAL
613                 **
614                 ** Purpose:
615                 **      Stores 15-digit form in registers A/B into buffer
616                 **      location pointed to by D1.
617                 **
618                 ** Entry:
619                 **      A/B - contains 15-digit form
620                 **      D1 -- point to 15-digit buffer location
621                 **
622                 ** Exit:
623                 **      buffer location has 15-digit form
624                 **      Carry=Clear
625                 **      P=14
626                 **
627                 ** Calls:      nothing
628                 **
629                 ** Uses.......
630                 **  Inclusive: CPU: P
631                 **             RAM: 21 nibbles: [D1,D1+20]
632                 **
633                 ** Stk lvls:   0
634                 **
635                 ** NOTE:
636                 **      See ADD for how this routine may be used.
637                 **
638                 ** History:
639                 **
640                 **    Date     Programmer              Modification
641                 **  --------   ----------   ---------------------------------
642                 **  06/10/82      PM         Documented routine
643                 **  01/06/83       "         Revised documentation
644                 **
645                 ********************************************************************
646                 ********************************************************************
647 0E170 2E    BFPTAB P=     14
648 0E172 A9C          ABEX   WP
649 0E175 1517         DAT1=A W              sign and mantissa
650 0E179 A9C          ABEX   WP
651 0E17C 17F          D1=D1+ 16
652 0E17F 141          DAT1=A A              exponent
653 0E182 1CF   D1-16  D1=D1- 16
654 0E185 01           RTN
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:    addsub  -  Adds or subtracts 1 from A/B
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Adds or subtracts 1 from a 15-digit form while
                **      preserving the meaning of SB to denote an inexact
                **      chain calculation.
                **
                ** Entry:
                **      standard 15-digit floating point math inputs
                **      sSTAT(s6)=0: add 1
                **               =1: subtract 1
                **
                ** Exit:
                **      see documentation for AD15S,SB15S
                **
                ** Calls:      falls into AD15S or SB15S
                **
                ** Uses.......
                **  Inclusive: A,B,C,D,P,SB,XM,sIX(s7)
                **
                ** Stk lvls:   1
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------   ----------   ----------------------------------
                **  01/06/83      PM         Documented routine
                **  03/04/83       "         Used sSTAT(s6) to sel. add or sub.
                **
                ********************************************************************
                ********************************************************************
 OE187 850    subone ST=1   =sSTAT
 OE18A AF2    addsub C=0    W
 OE18D AF3           D=0    W
 OE190 2E            P=     14
 OE192 B07           D=D+1  P
 OE195 860           ?ST=0  =sSTAT
 OE198 50            GOYES  AD15S
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) SB15S   -  15-digit subtract/add routine
                ** Name:(S) AD15S   -  15-digit subtract/add routine
                **
                ** Category:   MTHUTL
                **
                ** Purpose:
                **      Subtracts or adds, respectively, two 15-digit forms
                **      while preserving the meaning of SB to denote an inexact
```

```
710             **      chain calculation.
711             **
712             ** Entry:
713             **      A/B,C/D -- standard floating point math inputs
714             **      SB,XM ---- indicate prior inexact or invalid operation
715             **
716             ** Exit:
717             **      A/B ------ standard floating point math outputs
718             **      SB,XM ----
719             **      Carry set iff XM=1 on exit (e.g., Inf-Inf NaN created)
720             **
721             ** Calls:       AD15s,SAVESB,ORSB
722             **              exits through XMTEST
723             **
724             ** Uses.......
725             **  Inclusive: A,B,C,D,P,SB,XM,sIX(s7)
726             **
727             ** Stk lvls:   1
728             **
729             ** History:
730             **
731             **    Date      Programmer              Modification
732             **  --------    ----------   ------------------------------------
733             **  12/21/82       PM         Documented routine
734             **  01/31/83        "         Handled arithmetic-created NaNs
735             **
736             ********************************************************************
737             ********************************************************************
738 0E19A BCE   =SB15S  C=-C-1 S               needs dec mode
739 0E19D 8E00 =AD15S  GOSUBL =SAVESB          save SB in sIX
            00
740 0E1A3 8E00         GOSUBL =AD15s          needs dec mode
            00
741 0E1A9 8E00         GOSUBL =ORSB           update SB
            00
742 0E1AF 6B00         GOTO   XMTEST          set carry iff XM=1
743             ********************************************************************
744             ********************************************************************
745             **
746             ** Name:    mp15s   -  Connection to MP15S
747             **
748             ** Category:   LOCAL
749             **
750             ** Purpose:
751             **      Sets dec mode, jumps to MP15S
752             **
753             ** Entry/Exit:
754             **      See MP15S
755             **      Carry set iff XM=1 on exit (e.g. Inf*0 NaN created)
756             **
757             ** Uses.......
758             **  Inclusive: A,B,C,D,P,SB,XM
759             **
760             ** Stk lvls:   1
761             **
```

```
762             ********************************************************************
763             ********************************************************************
764 0E1B3 05    mp15s  SETDEC
765 0E1B5 8E00         GOSUBL =MP15S
          00
766             ********************************************************************
767 0E1BB 831   XMTEST ?XM=0                  Carry set iff XM=1
768 0E1BE 40           GOYES  rtncc2
769 0E1C0 02           RTNSC
770 0E1C2 03    rtncc2 RTNCC
```

```
771                     EJECT
772                 ****************************************************************
773                 ****************************************************************
774                 **
775                 ** Name:    ures12  -  Connection to uRES12
776                 **
777                 ** Category:   LOCAL
778                 **
779                 ** Purpose:
780                 **      Set P=IVP, move any error message into C(A), then
781                 **      fall into uRES12.
782                 **
783                 ** Entry:
784                 **      A/B -- 15-digit form
785                 **      D1 --- points to end of available memory
786                 **             (top of math stack)
787                 **
788                 ** Exit:
789                 **      uRES12 outputs
790                 **
791                 ** Calls:      GETMSG
792                 **             exits through uRES12
793                 **
794                 ** Uses.......
795                 **  Inclusive: A,B,C,D,P,R3,SB,XM,sIX(s7),s8-11
796                 **
797                 ** Stk lvls:   2 = uRES12
798                 **
799                 ** Note:
800                 **      This routine may fail to properly display a warning
801                 **      message if the message involves text insertion.
802                 **
803                 ** History:
804                 **
805                 **    Date     Programmer              Modification
806                 **  --------   ----------   ---------------------------------
807                 **  12/21/82      PM         Documented routine
808                 **  01/28/83       "         Revised documentation
809                 **
810                 ****************************************************************
811                 ****************************************************************
812 0E1C4 20     ures12 P=     =IVP            if exception, then is IV
813 0E1C6 7E50         GOSUB  GETMSG          put any message into C(A)
814 0E1CA 8C00         GOLONG =uRES12
          00
```

```
                        EJECT
                ****************************************************************
=RPTNaN ?XM=0
                        RTNYES
                        GONC    uRESD1          B.E.T.:  report Invalid NaN
                ****************************************************************
                ****************************************************************
                **
                ** Name:    MSARES  -  Chain calculation utility
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Multiply, read number from RAM, add, then fall into
                **      uRESD1.  Used for code packing.
                **
                ** Entry/Exit:
                **      See mp15s,spltA+,uRESD1
                **
                ** Calls:      AD15S,RCCD1X,STAB1,mp15s,spltA+
                **             exits through uRESD1
                **
                ** Uses.......
                **  Inclusive: CPU: A,B,C,D,P,R0-R1,R3,SB,XM,sIX(s7),s8-11
                **             RAM: STMTD1
                **
                ** Stk lvls:   5 = max(3,2+uRESD1)
                **
                ****************************************************************
                ****************************************************************
MSARES  GOSUB   mp15s
                *       GOC     uRESD1          jump if Inf*0 NaN created
                        GOSUBL  =STAB1          save product in R0/R1
                        GOSUB   spltA+          get, split, normalize, test
                        GOSUB   =RCCD1X         recall product into C/D
                ****************************************************************
                ****************************************************************
                **
                ** Name:    ARESD1  -  Chain calculation utility
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Adds two 15-digit forms and then falls into uRESD1.
                **      Used for code packing.
                **
                ** Entry/Exit:
                **      See AD15S,uRESD1
                **
                ** Uses.......
                **  Inclusive: see uRESD1
                **
                ** Stk lvls:   3
                **
```

```
869           ********************************************************************
870           ********************************************************************
871 0E1EA 7FAF  ARESD1 GOSUB  AD15S
872           ********************************************************************
873           ********************************************************************
874           **
875           ** Name:(S) uRESD1  -  Variation of uRES12
876           **
877           ** Category:   MTHUTL
878           **
879           ** Purpose:
880           **      Similar to uRES12.  Any XM exception is
881           **      considered an invalid operation (not a divide by zero
882           **      or 0^0 type).  AVMEME, rather than D1, points to the
883           **      end of available memory.  Also, various entities are
884           **      initialized on exit.
885           **
886           ** Entry:
887           **      A/B ----- 15-digit form for rounding, trap handling
888           **      XM ------ set iff invalid operation has occured
889           **      SB ------ set iff result is inexact
890           **      AVMEME -- points to end of available memory
891           **
892           ** Exit:
893           **      C ------- contains result
894           **      SB,XM = 0
895           **      s8-11 --- user modes
896           **      HEXMODE
897           **      Carry=Clear
898           **      P=14
899           **
900           ** Calls:      ures12
901           **             exits through uMODES
902           **
903           ** Uses.......
904           **  Inclusive:
905           **             CPU: A,B,C,D,P,R3,SB,XM,sIX(s7),s8-11
906           **             RAM: STMTD1
907           **
908           ** Stk lvls:   3 = 1+uRES12
909           **
910           ** Note:
911           **      This routine may fail to properly display a warning
912           **      message if the message involves text insertion.
913           **
914           ** History:
915           **
916           **    Date     Programmer            Modification
917           **  --------   ----------   ---------------------------------
918           **  06/10/82      PM        Documented routine
919           **  11/17/82       "        D1 preserved, falls through umode+
920           **  01/28/83       "        Fatal errors halt execution
921           **                            immediately
922           **
923           ********************************************************************
```

```
924             ********************************************************************
925 0E1EE 137   =uRESD1 CD1EX                  save D1 in STMTD1
926 0E1F1 1F00          D1=(5) =STMTD1
          000
927 0E1F8 145           DAT1=C A
928 0E1FB 1E00          D1=(4) =AVMEME         point D1 to AVMEME
          00
929 0E201 147           C=DAT1 A
930 0E204 135           D1=C
931 0E207 79BF          GOSUB  ures12
932 0E20B 1F00          D1=(5) =STMTD1         restore D1
          000
933 0E212 143           A=DAT1 A
934 0E215 131           D1=A
935 0E218 04    umodeH  SETHEX
936 0E21A 2E            P=     14
937 0E21C 822   umode+  SB=0
938 0E21F 821           XM=0
939 0E222 8C00          GOLONG =uMODES
          00
```

```
940                     EJECT
941                 ********************************************************************
942                 ********************************************************************
943                 **
944                 ** Name:    GETMSG  -  Get Signaling NaN Encoded Error Msg
945                 **
946                 ** Category:   LOCAL
947                 **
948                 ** Purpose:
949                 **      Puts error message encoded in NaN (A/B) into C(A).
950                 **
951                 ** Entry:
952                 **      A/B --- split NaN  (B(14-11) has error message code)
953                 **
954                 ** Exit:
955                 **      C(A) --- error message code
956                 **
957                 ** Calls:      nothing
958                 **
959                 ** Uses.......
960                 **  Inclusive: C
961                 **
962                 ** Stk lvls:   0
963                 **
964                 ** History:
965                 **
966                 **    Date     Programmer              Modification
967                 **  --------   ----------   ----------------------------------
968                 **  12/21/82      PM         Documented routine
969                 **  04/15/83       "         Messages in NaNs moved
970                 **
971                 ********************************************************************
972                 ********************************************************************
973 0E228 AF9   GETMSG C=B    W
974 0E22B 8C00          GOLONG =Cslc5
          00
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:    TEXPEX  -  Stacks list of n.e. on math stack
                **
                ** Category:   MTHSTK
                **
                ** Purpose:
                **      Collapses the math stack, then stacks a list of numeric
                **      expressions on it.
                **
                ** Entry:
                **      D0 -- points to numeric expression list (or statement
                **            terminator if there is no list)
                **
                ** Exit:
                **      math stack contains values of numeric expressions
                **      D0 ------ points to statement terminator
                **      D1 ------ points to top of math stack
                **      AVMEME -- points to top of math stack
                **      Carry=Set if no expression list
                **           =Clear otherwise
                **
                ** Calls:      COLLAP
                **             exits through EXPEXC
                **
                ** Uses.......
                **  Exclusive of EXPEXC: CPU: A(B),C(A),P,D1
                **                       RAM: MTHSTK/AVMEME
                **
                ** Stk lvls:   max(1,EXPEXC)
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **   --------   ----------   ----------------------------------
                **   06/10/82      PM         Documented routine
                **   01/06/83       "         Revised documentation
                **
                ****************************************************************
                ****************************************************************
OE231 8E00   TEXPEX GOSUBL =COLLAP        collapse math stack
          00
OE237 135           D1=C                  point D1 to MTHSTK/AVMEME
OE23A 14A           A=DAT0 B              test for no argument
OE23D 04            SETHEX
OE23F 21            P=     1
OE241 B04           A=A+1  P
OE244 400           RTNC                  no arg: return carry set
OE247 8C00   expexc GOLONG =EXPEXC           arg: return carry clear
          00
```

```
1025                    EJECT
1026 0E24D 6ED6  stscr  GOTO   STSCR
1027             ****************************************************************
1028             ****************************************************************
1029             **
1030             ** Name:     rclw1+  -  Save D0 then connect to RCLW1
1031             **
1032             ** Category:   LOCAL
1033             **
1034             ** Purpose:
1035             **       Save D0 in R3, then exit through RCLW1.
1036             **
1037             ** Entry:
1038             **       W1 -- value to be recalled from top of scratch math
1039             **             stack
1040             **       D0 -- pointer to be saved
1041             **
1042             ** Exit:
1043             **       R3 --- prior D0 value
1044             **       A/B -- recalled from top of scratch math stack
1045             **
1046             ** Calls:       GEXPAD,GSCPTR
1047             **
1048             ** Uses.......
1049             **  Inclusive: A,B,C,D,P,D0,R3
1050             **
1051             ** Stk lvls:   1
1052             **
1053             ** History:
1054             **
1055             **    Date      Programmer              Modification
1056             **  --------    ----------   ----------------------------------
1057             **  01/06/83      PM         Documented routine
1058             **
1059             ****************************************************************
1060             ****************************************************************
1061 0E251 136   rclw1+ CD0EX
1062 0E254 10B          R3=C
1063 0E257 6927  rclw1  GOTO   RCLW1          setdec
```

```
1064                    EJECT
1065                ********************************************************************
1066                ********************************************************************
1067                **
1068                ** Name:    pshstk  -  Pushes math stk
1069                **
1070                ** Category:   MTHSTK
1071                **
1072                ** Purpose:
1073                **      Pushes math stack 16 nibs, checking for out-of-memory
1074                **
1075                ** Entry:
1076                **      D1      @  Current top of mathstack
1077                **
1078                ** Exit:
1079                **      To NOMEM if insufficient memory, else return carry set
1080                **      D1      @  New top of mathstack, (pushed 16 nibs)
1081                **
1082                ** Calls:      STK16?, NOMEM
1083                **
1084                ** Uses.......
1085                **  Inclusive: C(A),D0,D1
1086                **
1087                ** Stk lvls:   1
1088                **
1089                ** History:
1090                **
1091                **    Date      Programmer              Modification
1092                **  --------    ----------   ----------------------------------
1093                **  11/22/82       PM         Documented routine
1094                **  01/06/83        "         Reviewed documentation
1095                **  09/29/83       FH         Documented entry/exit conditions
1096                **
1097                ********************************************************************
1098                ********************************************************************
1099 0E25B 132   pshstk ADOEX                    save A(A) in D0
1100 0E25E 8E00         GOSUBL =STK16?          increment math stack
           00
1101 0E264 132          ADOEX                   restore A(A)
1102 0E267 400          RTNC                    collision?
1103 0E26A 8C00         GOLONG =NOMEM           yes: fatal error
           00
```

```
1104                    EJECT
1105            ****************************************************************
1106            ****************************************************************
1107            **
1108            ** Name:    STFPRP  -  Statistical Function Preparation
1109            **
1110            ** Category:   LOCAL
1111            **
1112            ** Purpose:
1113            **      Statistical function preparation routine:  saves PC in
1114            **      function scratch, tests current statistical array,
1115            **      reads user modes, pops, tests, and converts variable
1116            **      number if present, otherwise pushes math stack and uses
1117            **      default variable number.
1118            **
1119            ** Entry:
1120            **      D0 ------- Program Counter
1121            **      STATAR --- current statistical array name
1122            **      optional variable number on top of math stack
1123            **
1124            ** Exit:
1125            **      Carry=Clear:
1126            **        A(A) ---- rounded hex variable number
1127            **        R2(S) --- # variables <=F
1128            **        R2(A) --- Address of SA(0)
1129            **        XM=0
1130            **        HEXMODE
1131            **        P=0
1132            **      Carry=Set:
1133            **        A/B ----- NaN
1134            **        XM=1 if NaN created
1135            **
1136            **
1137            ** Calls:      DCHXF,GETSDO,IVARG,POP1R,SPLTAX,pshstk,
1138            **             finita,umode+
1139            **
1140            ** Uses.......
1141            **  Inclusive: CPU: A,B,C,D(A),P,D0,R2,SB,XM,s8-11
1142            **                  also D1 if no arguments
1143            **             RAM: F-R0-0
1144            **
1145            ** Stk lvls:   3
1146            **
1147            ** History:
1148            **
1149            **    Date     Programmer            Modification
1150            **  --------   ----------   ----------------------------------
1151            **  12/21/82      PM         Documented routine
1152            **  03/24/83       "         Fatal error if no stat array
1153            **
1154            ****************************************************************
1155            ****************************************************************
1156 0E270 76D2  STFPRP GOSUB  GETSDO          save PC in F-R0-0
1157                                           get SA address, #VARS
1158 0E274 74AF         GOSUB  umode+          get user modes, set SB,XM to zero
```

```
1159 0E278 A4E          C=C-1  S
1160 0E27B 5D0          GONC   VARNBR          jump if 1 arg
1161 0E27E 79DF         GOSUB  pshstk          0 args: increment math stack.
1162 0E282 D0           A=0    A               use var. nbr. 1
1163 0E284 E4           A=A+1  A
1164 0E286 5C1          GONC   VAR=1           B.E.T.
```

```
1165                    EJECT
1166                ****************************************************************
1167                ****************************************************************
1168                **
1169                ** Name:(S) VARNBR  -  Pop and Test Variable Number
1170                ** Name:(S) VARNB-  -  Pop and Test Variable Number
1171                **
1172                ** Category:   CONVRT
1173                **
1174                ** Purpose:
1175                **      Rounds decimal floating point real value on top of
1176                **      math stack to a hex integer, then tests for a valid
1177                **      variable number.
1178                **      A NaN input will fall through; an out-of-range input
1179                **      will create a NaN -- both set carry.
1180                **
1181                ** Entry:
1182                **      decimal value to be converted on top of math stack
1183                **      D1 ----- points to top of math stack
1184                **      R2(S) -- # statistical variables
1185                **
1186                ** Exit:
1187                **      Carry=Set:   invalid input, NaN output in registers A/B
1188                **                   XM=1: If NaN created
1189                **      Carry=Clear: A(A) -- rounded hex integer
1190                **                   XM=0
1191                **                   HEXMODE
1192                **                   P=0
1193                **
1194                ** Calls:      DCHXF,IVARG,POP1R,SPLTAX,finita
1195                **
1196                ** Uses.......
1197                **  Inclusive: VARNB-: A,B,C,P,XM
1198                **             VARNBR: same, unless fatal error
1199                **
1200                ** Stk lvls:   2
1201                **
1202                ** History:
1203                **
1204                **     Date     Programmer           Modification
1205                **   --------   ----------   --------------------------------
1206                **   06/09/82      PM        Documented routine
1207                **   08/12/82       "        Changed entry points
1208                **   12/14/82       "        Added signaling NaN test
1209                **   02/10/83       "        Fixed neg var nbr problem
1210                **
1211                ****************************************************************
1212                ****************************************************************
1213 0E289 7076 =VARNBR GOSUB  POP1R          pop real argument
1214 0E28D 7596 =VARNB- GOSUB  finita         detect nonfinite, don't normalize
1215 0E291 402          GOC    IVVRNB         nonfinite? (preserve NaNs)
1216 0E294 8F00         GOSBVL =DCHXF         round and convert to hex
           000
1217 0E29B 470          GOC    VAR=1          jump if positive and no overflow
1218 0E29E 8AC          ?A#0   A              <0 or overflow?
```

```
1219 0E2A1 C1           GOYES  aornan
1220 0E2A3 11A  VAR=1   C=R2
1221 0E2A6 D2           C=0    A
1222 0E2A8 812          CSLC
1223 0E2AB 8B6          ?A>C   A              var nbr > NVAR ?
1224 0E2AE F0           GOYES  aornan
1225 0E2B0 03           RTNCC                 legal var nbr: return with CClear
1226 0E2B2 968  IVVRNB  ?A=0   B              Inf?
1227 0E2B5 80           GOYES  aornan
1228 0E2B7 7073         GOSUB  SPLTAX         no: split,test for signan
1229 0E2BB 02           RTNSC
1230 0E2BD 8E00 aornan  GOSUBL =IVARG         create NaN, set XM, carry
          00
1231 0E2C3 02           RTNSC
```

```
1232                    EJECT
1233                ****************************************************************
1234                ****************************************************************
1235                **
1236                ** Name:     TOTAL   -  TOTAL, MEAN Function Execute
1237                ** Name:     MEAN    -  TOTAL, MEAN Function Execute
1238                **
1239                ** Category:   FNEXEC
1240                **
1241                ** Purpose:
1242                **      Returns the sample total and mean, respectively, of
1243                **      specified variable in the current statistical array.
1244                **
1245                ** Entry:
1246                **      Carry=Clear
1247                **      variable # I on top of math stack
1248                **      current statistical array available
1249                **      C(S) -- argument count
1250                **
1251                ** Exit:
1252                **      total or mean on top of math stack
1253                **
1254                ** Calls:       TOTAL: RSTDO,STFPRP,TBLTOT,ures12
1255                **              MEAN:  also SGNPRD
1256                **              exits through FNRTN4
1257                **
1258                ** Uses.......
1259                **  Inclusive:
1260                **      TOTAL: CPU: A,B,C,D,P,R2,R3,SB,XM,sSTAT(s6),sIX(s7),
1261                **                  s8-11, also D1 if no argument
1262                **             RAM: F-R0-0
1263                **       MEAN: also D, scratch math stack
1264                **
1265                ** Stk lvls:   4 = max(4,1+uRES12)
1266                **
1267                ** History:
1268                **
1269                **     Date     Programmer           Modification
1270                **   --------   ----------   ---------------------------------
1271                **   06/09/82      PM         Documented routine
1272                **   08/13/82       "         Merged TOTAL,MEAN
1273                **   11/19/82       "         Fixed no parameter case
1274                **   12/17/82       "         Saved PC in function scratch
1275                **   01/31/83       "         Handled arithmetic-created NaNs
1276                **   03/24/83       "         Fatal error if no stat array
1277                **
1278                ****************************************************************
1279                ****************************************************************
1280 0E2C5 801          NIBHEX 801
1281 0E2C8 840  =TOTAL  ST=0   =sSTAT
1282 0E2CB 580          GONC   MEAN1          B.E.T.
1283 0E2CE 801          NIBHEX 801
1284 0E2D1 850  =MEAN   ST=1   =sSTAT
1285 0E2D4 789F  MEAN1  GOSUB  STFPRP         save PC (D0) in F-R0-0
1286                                          get SA address, #VARS
```

```
1287                                          push stack if NaN and no args
1288                                          pop, round, test I; use 1 if no arg
1289 0E2D8 4F0          GOC    MEAN3         NaN created?
1290 0E2DB 7023         GOSUB  TBLTOT        get T(0,I)
1291           *         GOC    MEAN3         Signaling NaN?
1292 0E2DF 860          ?ST=0  =sSTAT        TOTAL?
1293 0E2E2 60           GOYES  MEAN3
1294 0E2E4 7E02         GOSUB  SGNPRD        W1:=T(0,I), get T(0,0)=N
1295                                          T(0,I)/T(0,0) or NaN
1296 0E2E8 6AE0  MEAN3  GOTO   CORR3
```

```
1297                    EJECT
1298                ****************************************************************
1299                ****************************************************************
1300                **
1301                ** Name:    SDEV    -  Standard Deviation Function
1302                **
1303                ** Category:   FNEXEC
1304                **
1305                ** Purpose:
1306                **      Computes the standard deviation of the specified
1307                **      variable in the current statistical array.
1308                **      This is a sample estimate of the population parameter.
1309                **
1310                ** Entry:
1311                **      variable # I on top of math stack
1312                **      current statistical array available
1313                **      C(S) -- argument count
1314                **
1315                ** Exit:
1316                **      estimate on top of math stack
1317                **
1318                ** Calls:      NNGTAB,POSTAB,PTRCDV,RSTDO,SQRSAV,STFPRP,
1319                **             ST&GTN,ZATBLU,subone,ures12,
1320                **             exits through FNRTN4
1321                **
1322                ** Uses.......
1323                **  Inclusive:
1324                **        CPU: A,B,C,D,P,R2,R3,SB,XM,sSTAT(s6),sIX(s7),s8-11
1325                **             (,D1 if no argument)
1326                **        RAM: scratch math stack, F-R0-0
1327                **
1328                ** Stk lvls:   4 = max(4,1+uRES12)
1329                **
1330                ** History:
1331                **
1332                **     Date     Programmer            Modification
1333                **    --------   ----------   ---------------------------------
1334                **    06/03/82      PM         Documented routine
1335                **    12/17/82       "         Saved PC in function scratch
1336                **    01/31/83       "         Handled arithmetic-created NaNs
1337                **    03/24/83       "         Fatal error if no stat array
1338                **
1339                ****************************************************************
1340                ****************************************************************
1341 0E2EC 801            NIBHEX 801
1342 0E2EF 7D7F =SDEV     GOSUB  STFPRP          save PC (D0) in F-R0-0
1343                                              get SA address, #VARS
1344                                              push stack if NaN and no args
1345                                              pop, round, test I; use 1 if no arg
1346 0E2F3 422            GOC    SDEV2           NaN created?
1347 0E2F6 D6             C=A    A
1348 0E2F8 70F2           GOSUB  ZATBLU          get T(I,I). Return 0 if I=0. setdec
1349              *        GOC    SDEV2           Signaling NaN?
1350 0E2FC 7612           GOSUB  NNGTAB          valid sample?
1351              *        GOC    SDEV2           NaN created?
```

```
1352 0E300 75F2          GOSUB  ST&GTN        W1:=T(I,I), get T(0,0)=N
1353            *        GOC    SDEV2         Signaling NaN?
1354 0E304 7402          GOSUB  POSTAB        insure proper message when N=0
1355            *        GOC    SDEV2
1356 0E308 7B7E          GOSUB  subone        N-1
1357 0E30C 7AE1          GOSUB  PTRCDV        valid sample/stat oper? N>1?
1358                                           Inf/Inf?
1359 0E310 8E00          GOSUBL =SQRSAV       SQR(T(I,I)/(N-1))
           00
1360 0E316 6CB0  SDEV2   GOTO   CORR3         restore PC
```

```
1361                    EJECT
1362                ********************************************************************
1363                ********************************************************************
1364                **
1365                ** Name:    PREDV   -  Predicted Value Function
1366                **
1367                ** Category:   FNEXEC
1368                **
1369                ** Purpose:
1370                **      Computes the predicted value for the current regression
1371                **      and dependent variable value.
1372                **
1373                ** Entry:
1374                **      dependent variable value on top of math stack
1375                **      dependent, independent variable #'s in stat array's
1376                **                dope vector
1377                **
1378                ** Exit:
1379                **      predicted value on top of math stack
1380                **
1381                ** Calls:      RD15S,ARGPRP,GETSDO,GTINSL,RCLW1,RCLW3,
1382                **             RSTDO,STSCR,mp15s,ures12
1383                **             exits through FNRTN4
1384                **
1385                ** Uses.......
1386                **  Inclusive:
1387                **        CPU: A,B,C,D,P,R0-3,SB,XM,sIX(s7),s8-11
1388                **        RAM: scratch math stack, F-R0-0
1389                **
1390                ** Stk lvls:   4 = max(4,1+uRES12)
1391                **
1392                ** History:
1393                **
1394                **     Date     Programmer            Modification
1395                **   --------   ----------   ---------------------------------
1396                **   06/03/82      PM         Documented routine
1397                **   12/17/82       "         Saved PC in function scratch
1398                **   01/31/83       "         Handled arithmetic-created NaNs
1399                **   03/24/83       "         Fatal error if no stat array
1400                **
1401                ********************************************************************
1402                ********************************************************************
1403 0E31A 811           NIBHEX 811
1404 0E31D 7922 =PREDV   GOSUB  GETSDO          save PC (D0) in F-R0-0
1405                                             get SA address, #VAR, P:=0
1406 0E321 D2            C=0    A
1407 0E323 F5            BSR    A
1408 0E325 A89           C=B    P
1409 0E328 109           R1=C                    R1:=J=indep var #
1410 0E32B F5            BSR    A
1411 0E32D A89           C=B    P
1412 0E330 108           R0=C                    R0:=I=dep var #
1413 0E333 7281          GOSUB  GTINSL           get INTCPT, SLOPE, user modes
1414              *      GOC    PREDV1           NaN created?
1415 0E337 71F5          GOSUB  STSCR
```

```
1416 0E33B 70B5          GOSUB  ARGPRP        pop, prepare x
1417             *       GOC    PREDV1        signaling NaN?
1418 0E33F 7186          GOSUB  RCLW3
1419 0E343 7C6E          GOSUB  mp15s         SLOPE*x
1420             *       GOC    PREDV1        Inf*0 NaN?
1421 0E347 7636          GOSUB  RCLW1
1422 0E34B 7E4E          GOSUB  AD15S         INTCPT + SLOPE*x
1423                                           (Inf-Inf NaN caught)
1424 0E34F 6380  PREDV1  GOTO   CORR3
```

```
1425                    EJECT
1426                ****************************************************************
1427                ****************************************************************
1428                **
1429                ** Name:    CORR    -  Correlation Function
1430                **
1431                ** Category:   FNEXEC
1432                **
1433                ** Purpose:
1434                **      Computes the correlation coefficient between the two
1435                **      specified variables in the current statistical array.
1436                **
1437                ** Entry:
1438                **      variables #'s I,J on top of math stack
1439                **      current statistical array defined
1440                **
1441                ** Exit:
1442                **      correlation coefficient on top of stack
1443                **
1444                ** Calls:      GETSDO,MSN12,POSTAB,RCLW1,RCSCR,RSTDO,
1445                **             SQRSAV,STSCR,TBLU,VARNB-,ZATBLU,dv15s,
1446                **             mp15s,pop2n+,umode+,ures12
1447                **             exits through FNRTN4
1448                **
1449                ** Uses.......
1450                **  Inclusive:
1451                **        CPU: A,B,C,D,P,D1,R0-3,SB,XM,sIX(s7),s8-11
1452                **        RAM: scratch math stack,F-R0-0
1453                **
1454                ** Stk lvls:   4 = max(3,2+uRES12)
1455                **
1456                **
1457                ** History:
1458                **
1459                **     Date     Programmer           Modification
1460                **   --------   ----------   ---------------------------------
1461                **   06/03/82      PM        Documented routine
1462                **   12/17/82       "        Saved PC in function scratch
1463                **   01/11/83       "        Changed order of tests, added
1464                **                             most significant NaN test.
1465                **   01/12/83       "        Pop 1st argument if no valid
1466                **                             current statistical array
1467                **   01/17/83       "        Replaced pop1n+ with pop2n+
1468                **   01/31/83       "        Handled arithmetic-created NaNs
1469                **   03/24/83       "        Fatal error if no stat array
1470                **
1471                ****************************************************************
1472                ****************************************************************
1473 0E353 62B5  cnflc2 GOTO   cnflct
1474 0E357 8822         NIBHEX 8822
1475 0E35B 7BE1 =CORR   GOSUB  GETSDO          save PC (D0) in F-R0-0
1476                                           get SA address, #VARS, P:=0
1477 0E35F 7495         GOSUB  pop2n+          pop 2 real args (signan test)
1478 0E363 4FE          GOC    cnflc2
1479 0E366 100          R0=A                    R0:=I
```

```
1480 0E369 109          R1=C                R1:=J
1481 0E36C 7CAE         GOSUB  umode+       get user modes, clear SB,XM
1482 0E370 8E00         GOSUBL =MSN12       find most significant NaN
           00
1483 0E376 4C5          GOC    CORR3        NaN?
1484 0E379 110          A=R0
1485 0E37C 7D0F         GOSUB  VARNB-       split, round, test I; sethex
1486 0E380 425          GOC    CORR3        NaN?
1487 0E383 121          AR1EX               R1:=I
1488 0E386 730F         GOSUB  VARNB-       split, round, test J
1489 0E38A 484          GOC    CORR3        NaN?
1490 0E38D 100          R0=A                R0:=J
1491 0E390 119          C=R1
1492 0E393 7A62         GOSUB  TBLU         get T(I,J), setdec
1493             *      GOC    CORR3        Signaling NaN?
1494 0E397 7195         GOSUB  STSCR        W1:=T(I,J)
1495 0E39B 118          C=R0
1496 0E39E 7A42         GOSUB  ZATBLU       get T(J,J). Return 0 if J=0
1497             *      GOC    CORR3        Signaling NaN?
1498 0E3A2 7661         GOSUB  POSTAB       valid sample/stat oper? T(J,J)>0?
1499             *      GOC    CORR3        NaN created?
1500 0E3A6 7285         GOSUB  STSCR        W1:=T(J,J), W2=T(I,J)
1501 0E3AA 119          C=R1
1502 0E3AD 7B32         GOSUB  ZATBLU       get T(I,I). Return 0 if I=0
1503             *      GOC    CORR3        signaling NaN?
1504 0E3B1 7751         GOSUB  POSTAB       valid sample/stat oper? T(I,I)>0?
1505             *      GOC    CORR3        NaN created?
1506 0E3B5 7B95         GOSUB  RCSCR        W1=T(I,J)
1507 0E3B9 76FD         GOSUB  mp15s        (Inf*0 not possible)
1508 0E3BD 8E00         GOSUBL =SQRSAV      den=SQR(T(I,I)*T(J,J))>0
           00
1509 0E3C3 7AB5         GOSUB  RCLW1
1510 0E3C7 95D          ?B#0   M            clear SB if T(I,J)=0
1511 0E3CA 50           GOYES  CORR2
1512 0E3CC 822          SB=0
1513 0E3CF 7F21  CORR2  GOSUB  dv15s        CORR=T(I,J)/den
1514                                        Inf/Inf? (0/0,DVZ not possible)
1515 0E3D3 8E00  CORR3  GOSUBL =RSTD0       restore PC (D0)
           00
1516 0E3D9 77ED  outres GOSUB  ures12
1517 0E3DD 8C00         GOLONG =FNRTN4
           00
```

```
1518                EJECT
1519            ****************************************************************
1520            ****************************************************************
1521            **
1522            ** Name:     LR      -  Simple linear regression
1523            **
1524            ** Category:   STEXEC
1525            **
1526            ** Purpose:
1527            **      Stores the dependent and independent variable numbers
1528            **      in the current statistical array's dope vector.  If a
1529            **      variable name list is included, the intercept and
1530            **      slope of the regression are stored in the specified
1531            **      variables, respectively.
1532            **
1533            ** Entry:
1534            **      dependent, independent variable numbers parsed
1535            **
1536            ** Exit:
1537            **      dope vector nib 10 -- dependent variable number
1538            **      dope vector nib  9 -- independent variable number
1539            **      slope and intercept in optional variable(s)
1540            **      Fatal error if bad var # or no valid stat array
1541            **
1542            ** Calls:       ADRS5O,B=STAN,DEST,D1STOR,D1=AVE,GETSA,
1543            **              GTINSL,POPMTH,RCLW2,SVTRC,TEXPEX,VARNBR,
1544            **              ave=d1,expexc,umode+,ures12
1545            **              exits through nxtstm
1546            **
1547            ** Uses.......
1548            **  Exclusive of EXPEXC,STORE:
1549            **        CPU: A,B,C,D,P,DO,D1,RO-3,SB,XM,sIX(s7),s8-11
1550            **        RAM:  math scratch stack
1551            **              S-RO-0,1,2,3 (DEST)
1552            **              S-R1-0,1,2,3 (DEST)
1553            **              MTHSTK,AVMEME
1554            **
1555            ** Stk lvls:   max(5,1+EXPEXC)
1556            **
1557            ** History:
1558            **
1559            **     Date     Programmer           Modification
1560            **   --------   ----------   ---------------------------------
1561            **   06/02/82      PM         Documented routine
1562            **   08/27/82       "         Always computes parameters
1563            **   11/22/82       "         Created GTINSL routine
1564            **   11/30/82       "         Fatal error for invalid var #
1565            **   12/16/82       "         Allowed trace
1566            **   01/31/83       "         Handled arithmetic-created NaNs
1567            **   03/24/83       "         Fatal error if no stat array
1568            **   05/10/83       "         Stacked INTCPT,SLOPE on math
1569            **                               stack for variable STORE
1570            **
1571            ****************************************************************
1572            ****************************************************************
```

```
1573 0E3E3 6C35  LRIVAE GOTO   IVAERR          fatal error: invalid argument
1574 0E3E7 79DD  LRSKIP GOSUB  ures12          set IVL flag, consult IVL trap
1575 0E3EB 10A           R2=C                  put created NaN into R2,R0 and
1576 0E3EE 6860          GOTO   LR1              continue
1577 0E3F2 0000          REL(5) =LRDC
           0
1578 0E3F7 0000          REL(5) =LRP
           0
1579 0E3FC 713E =LR      GOSUB  TEXPEX         build expression stack, point D1
1580 0E400 136           CDOEX                 save PC
1581 0E403 06            RSTK=C
1582 0E405 7841          GOSUB  GETSA          get SA address, #VARS, test length
1583 0E409 7C7E          GOSUB  VARNBR         pop, round, test J; sethex
1584 0E40D 45D           GOC    LRIVAE         invalid argument error?
1585 0E410 101           R1=A                  R1:=J=indep var # in hex
1586 0E413 17F           D1=D1+ 16
1587 0E416 7F6E          GOSUB  VARNBR         pop, round, test I; sethex
1588 0E41A 48C           GOC    LRIVAE         invalid argument error?
1589 0E41D 100           R0=A                  R0:=I=dep var #  in hex
1590 0E420 7DA1          GOSUB  B=STAN         read current stat array name
1591 0E424 8E00          GOSUBL =ADRS50        get dope vector address
           00
1592 0E42A 163           D0=D0+ 4              point to dep,indep var #'s
1593 0E42D 118           C=R0
1594 0E430 F2            CSL    A
1595 0E432 DA            A=C    A              update dep var #
1596 0E434 119           C=R1
1597 0E437 A8A           A=C    P              update indep var #
1598 0E43A 148           DAT0=A B
1599 0E43D 7870          GOSUB  GTINSL         get INTCPT,SLOPE
1600 0E441 45A           GOC    LRSKIP         NaN created?
1601 0E444 7C7D          GOSUB  ures12         fatal error halts here!!
1602                                           (D1 points to top of math stack)
1603 0E448 70DD          GOSUB  umode+         reread user modes, etc
1604 0E44C 10A           R2=C                  R2=INTCPT,W2=SLOPE
1605 0E44F 7B65          GOSUB  RCLW2
1606 0E453 7D6D          GOSUB  ures12         fatal error halts here!!
1607                                           (D1 points to top of math stack)
1608 0E457 108   LR1     R0=C                  R0=SLOPE,R2=INTCPT
1609 0E45A 07            C=RSTK                restore PC
1610 0E45C 136           CDOEX
1611 0E45F 110           A=R0                  push SLOPE onto math stack
1612 0E462 1517          DAT1=A W
1613 0E466 112           A=R2                  push INTCPT onto math stack
1614 0E469 1CF           D1=D1- 16
1615 0E46C 1517          DAT1=A W
1616             ************************************************************
1617             * Here:  D1     --> INTCPT (16 nibs)                       *
1618             *                   SLOPE  (16 nibs)                       *
1619             *        FORSTK -->                                        *
1620             *                                                          *
1621             ************************************************************
1622 0E470 8E00  LRVRLP GOSUBL =ave=d1
           00
1623             ****************************************************************
```

```
1624                    * Here:  AVMEME,D1 --> value to be stored on top of stack     *
1625                    *                                                             *
1626                    * Now test for presence of destination variable name:        *
1627                    ***************************************************************
1628 0E476 14A                 A=DAT0 B
1629 0E479 20                  P=     0
1630 0E47B 311F                LCHEX  F1              load comma token
1631 0E47F 966                 ?A#C   B
1632 0E482 33                  GOYES  LREND
1633                    ***************************************************************
1634                    * Name present: find address, etc.                           *
1635                    ***************************************************************
1636 0E484 161                 D0=D0+ 2               skip over comma token
1637 0E487 8E00                GOSUBL =SVTRC          save PC for TRACE
           00
1638 0E48D 76BD                GOSUB  expexc          get variable name
1639 0E491 8E00                GOSUBL =DEST           save destination variable name
           00
1640                    ***************************************************************
1641                    * Repoint AVMEME,D1 to value to be stored into destination    *
1642                    * by finding and popping top entry off the math stack.        *
1643                    ***************************************************************
1644 0E497 20                  P=     0
1645 0E499 8F00                GOSBVL =D1=AVE         point D1 to top stack entry
           000
1646 0E4A0 8F00                GOSBVL =POPMTH         pop top entry off stack (AB&UTL)
           000
1647                    ***************************************************************
1648                    * Put new D1 into AVMEME.  Then store the value into the      *
1649                    * destination variable.  Handle TRACE VARS if it is in effect *
1650                    ***************************************************************
1651 0E4A7 8F00                GOSBVL =D1STOR         D1 is preserved  (SB&IO)
           000
1652                    ***************************************************************
1653                    * Increment D1 to point at next value to be stored and loop   *
1654                    * back to check for another desination variable.              *
1655                    ***************************************************************
1656 0E4AE 17F                 D1=D1+ 16
1657 0E4B1 6EBF                GOTO   LRVRLP
1658 0E4B5 619B  LREND  GOTO   nxtstm
```

```
1659                    EJECT
1660                ****************************************************************
1661                ****************************************************************
1662                **
1663                ** Name:    GTINSL  -  Compute slope and intercept for LR
1664                **
1665                ** Category:   LOCAL
1666                **
1667                ** Purpose:
1668                **      Computes the slope and intercept for the current simple
1669                **      regression.
1670                **
1671                ** Entry:
1672                **      R0(A) --- dependent variable number (hex)
1673                **      R1(A) --- independent variable number (hex)
1674                **      SB,XM --- no meaning
1675                **
1676                ** Exit:
1677                **      A/B --- Intercept (INTCPT) in 15-digit form
1678                **      W2 ---- Slope in 15-digit form
1679                **      SB ---- set/clear as appropriate
1680                **      Carry set iff XM set
1681                **
1682                ** Calls:      DV15S,POSTAB,PTRCDV,RCLW1,RCSCR,SB15S,STSCR,
1683                **             ST&GTN,TBLTOT,ZATBLU,ZTBLU+,mp15s,umode+
1684                **
1685                ** Uses.......
1686                **  Inclusive: A,B,C,D,P,D0,SB,XM,sIX(s7),s8-11
1687                **             scratch math stack
1688                **
1689                ** Stk lvls:   3
1690                **
1691                ** History:
1692                **
1693                **    Date      Programmer            Modification
1694                **  --------    ----------   ---------------------------------
1695                **  11/22/82       PM         Constructed from LR routine code
1696                **  01/31/82        "         Carry set iff XM=1 on exit
1697                **  01/31/83        "         Handled arithmetic-created NaNs
1698                **
1699                ****************************************************************
1700                ****************************************************************
1701 0E4B9 7F5D  GTINSL GOSUB  umode+          get user modes, set SB,XM=0
1702 0E4BD 7021        GOSUB  ZTBLU+          get T(J,I). If I=0 then SLOPE=0
1703                *      RTNC               Signaling NaN?
1704 0E4C1 7764        GOSUB  STSCR           W1:=T(J,I),R0=I,R1=J
1705 0E4C5 119         C=R1
1706 0E4C8 7021        GOSUB  ZATBLU          get T(J,J). Returns 0 if J=0.
1707                *      RTNC               Signaling NaN?
1708 0E4CC 7A20        GOSUB  PTRCDV          valid sample/stat oper? Inf/Inf?
1709                *      RTNC               NaN created?
1710 0E4D0 7854        GOSUB  STSCR           W1:=SLOPE=T(J,I)/T(J,J),R0=I,R1=J
1711 0E4D4 111         A=R1
1712 0E4D7 7421        GOSUB  TBLTOT          get T(0,J), sets hex
1713                *      RTNC               Signaling NaN?
```

```
1714 0E4DB 72A4          GOSUB  RCLW1
1715 0E4DF 70DC          GOSUB  mp15s          SLOPE*T(0,J)
1716              *      RTNC                  Inf*0 NaN created?
1717 0E4E3 7544          GOSUB  STSCR          W1=SLOPE*T(0,J),W2=SLOPE,R0=I,R1=J
1718 0E4E7 110           A=R0
1719 0E4EA 7111          GOSUB  TBLTOT         get T(0,I)
1720              *      RTNC                  Signaling NaN?
1721 0E4EE 7264          GOSUB  RCSCR
1722 0E4F2 74AC          GOSUB  SB15S
1723              *      RTNC                  Inf-Inf NaN created?
1724              ********************************************************************
1725              ********************************************************************
1726              **
1727              ** Name:    SGNPRD  -  Chain calculation utility
1728              **
1729              ** Category:   LOCAL
1730              **
1731              ** Purpose:
1732              **      Pushes 15-digit numerator on top of math stack, gets
1733              **      SA(0) (denominator) and tests that is not <=0, recalls
1734              **      numerator and computes quotient.  If SA(0) is a
1735              **      signaling NaN or is <=0, a NaN is created.
1736              **
1737              ** Entry:
1738              **      A/B --- 15-digit numerator
1739              **
1740              ** Exit:
1741              **      A/B -- quotient
1742              **      SB --- set/clear as appropriate
1743              **      W1 --- numerator
1744              **      Carry=Set iff XM set
1745              **
1746              ** Calls:      DV15S,POSTAB,RCLW1,ST&GTN
1747              **
1748              ** Uses.......
1749              **  Inclusive: A,B,C,D,P,SB,XM,D0
1750              **             scratch math stack
1751              **
1752              ** Stk lvls:   3
1753              **
1754              ** History:
1755              **
1756              **    Date     Programmer             Modification
1757              **  --------   ----------   ---------------------------------
1758              **  12/21/82      PM         Documented routine
1759              **  01/31/83       "         Carry set iff XM=1 on exit
1760              **
1761              ********************************************************************
1762              ********************************************************************
1763 0E4F6 7FF0  SGNPRD GOSUB  ST&GTN          W1=T(0,I)-W1,W2=SLOPE,R0=I,R1=J,
1764                                             get T(0,0)=N
1765              *      RTNC                  Signaling NaN?
1766              ********************************************************************
1767              ********************************************************************
1768              **
```

```
1769            ** Name:    PTRCDV  -  Test denominator, recall, divide
1770            **
1771            ** Category:   LOCAL
1772            **
1773            ** Purpose:
1774            **      Tests that 15-digit denominator is not <=0.  If not,
1775            **      recalls numerator and computes quotient.  Otherwise,
1776            **      creates a NaN.
1777            **
1778            ** Entry:
1779            **      A/B --- 15-digit denominator
1780            **
1781            ** Exit:
1782            **      A/B --- quotient
1783            **      SB ---- set/clear as appropriate
1784            **      Carry=Set iff XM set
1785            **
1786            ** Calls:      DV15S,POSTAB,RCLW1
1787            **             exits through XMTEST
1788            **
1789            ** Uses.......
1790            **  Inclusive: A,B,C,D,P,SB,XM,DO
1791            **
1792            ** Stk lvls:   2
1793            **
1794            ** History:
1795            **
1796            **    Date     Programmer             Modification
1797            **  --------   ----------   ----------------------------------
1798            **  12/21/82      PM         Documented routine
1799            **  01/31/83       "         Set carry iff XM=1 on exit
1800            **
1801            ********************************************************************
1802            ********************************************************************
1803 0E4FA 7E00  PTRCDV GOSUB  POSTAB         valid sample/stat oper?
1804            *       RTNC                  NaN created?
1805 0E4FE 7F74        GOSUB  RCLW1
1806 0E502 8E00  dv15s  GOSUBL =DV15S         INTCPT:=W1/N  (needs dec mode)
           00
1807 0E508 62BC        GOTO   XMTEST         Set carry iff XM is set
```

```
1808                    EJECT
1809            ****************************************************************
1810            ****************************************************************
1811            **
1812            ** Name:    POSTAB  -  Tests against <= 0
1813            **
1814            ** Category:   MTHUTL
1815            **
1816            ** Purpose:
1817            **      Tests that the 15-digit value in registers A/B is not
1818            **      <=0.  If it is <=0, a NaN is generated and carry is set.
1819            **
1820            ** Entry:
1821            **      A/B -- 15-digit form
1822            **
1823            ** Exit:
1824            **      NaN: Carry=Clear
1825            **       >0: Carry=Clear
1826            **           DECMODE
1827            **       =0: eIVSOP NaN created
1828            **           Carry=Set, XM=1
1829            **       <0: eIVSTA NaN created
1830            **           Carry=Set, XM=1
1831            **
1832            ** Calls:      INVNaN(if NaN created),finita
1833            **
1834            ** Uses.......
1835            **  Inclusive: if NaN created: A,B,C(A),P,XM
1836            **                  otherwise: nothing
1837            **
1838            ** Stk lvls:   1
1839            **
1840            ** History:
1841            **
1842            **    Date     Programmer             Modification
1843            **  --------   ----------   ---------------------------------
1844            **  06/10/82      PM         Documented routine
1845            **  01/06/83       "         Reviewed documentation
1846            **
1847            ****************************************************************
1848            ****************************************************************
1849 0E50C 95D   POSTAB ?B#0   M              test for =0
1850 0E50F 70          GOYES  NNGTAB
1851 0E511 8A8         ?A=0   A
1852 0E514 12          GOYES  IVSOPR
1853            ****************************************************************
1854            ****************************************************************
1855            **
1856            ** Name:    NNGTAB  -  Tests against < 0
1857            **
1858            ** Category:   MTHUTL
1859            **
1860            ** Purpose:
1861            **      Tests that the 15-digit value in registers A/B is not
1862            **      <0.  If it is <0, a NaN is created and carry set.
```

```
1863            **      See POSTAB documentation for more information.
1864            **
1865            ****************************************************************
1866            ****************************************************************
1867 0E516 7C04  NNGTAB GOSUB  finita          sets dec mode
1868 0E51A 532         GONC   NNGFIN
1869 0E51D 96C         ?A#0   B               NaN?
1870 0E520 82          GOYES  PRTNCC
1871 0E522 948         ?A=0   S               +inf?
1872 0E525 32          GOYES  PRTNCC
1873 0E527 20    IVSTAT P=     0               invalid statistic:
1874 0E529 3100        LC(2)  =eIVSTA
1875            ****************************************************************
1876            ****************************************************************
1877            **
1878            ** Name:    invnan  -  Create a NaN With Encoded Error Msg
1879            **
1880            ** Category:   MTHUTL
1881            **
1882            ** Purpose:
1883            **      Create a NaN, set XM, set Carry
1884            **
1885            ** Entry:
1886            **      C(B) -- mainframe error message code
1887            **
1888            ** Exit:
1889            **      A/B --- split NaN
1890            **      C(A) -- 5 nib message
1891            **      P ----- IVP
1892            **
1893            ** Calls:      INVNaN
1894            **
1895            ** Uses.......
1896            **  Inclusive: A,B,C(A),P,XM
1897            **
1898            ** Stk lvls:   1
1899            **
1900            ** History:
1901            **
1902            **    Date     Programmer             Modification
1903            **  --------   ----------   ----------------------------------
1904            **  12/21/82      PM        Documented routine
1905            **  01/06/83       "        Reviewed documentation
1906            **
1907            ****************************************************************
1908            ****************************************************************
1909 0E52D 8E00  invnan GOSUBL =INVNaN        create a NaN, set XM
          00
1910 0E533 02          RTNSC                  set carry
1911            ****************************************************************
1912 0E535 20    IVSOPR P=     0               +-0: invalid statistical operation
1913 0E537 3100        LC(2)  =eIVSOP
1914 0E53B 41F         GOC    invnan          B.E.T.
1915 0E53E 959   NNGFIN ?B=0   M               finite: treat -0 as 0
1916 0E541 70          GOYES  PRTNCC
```

```
1917 0E543 94C          ?A#0   S              <0?
1918 0E546 1E           GOYES  IVSTAT
1919 0E548 03    PRTNCC RTNCC
```

```
1920                EJECT
1921            ****************************************************************
1922            ****************************************************************
1923            **
1924            ** Name:(S) GETSA   -  Tests current statistical array
1925            ** Name:    GETSDO  -  Tests current statistical array
1926            **
1927            ** Category:   MTHUTL
1928            **
1929            ** Purpose:
1930            **      Gets the starting address of the current statistical
1931            **      array, to record and test the number of variables, and
1932            **      to test the length of this array.  GETSDO does the
1933            **      same after saving DO in function scratch.
1934            **
1935            ** Entry:
1936            **      Current statistical array name stored at =STATAR
1937            **
1938            ** Exit:
1939            **     Carry=Clear:
1940            **       A(S) --- # variables <=F
1941            **       A(A) --- address of first element of current stat array
1942            **       B(X) --- depend var#, independ var#, #variables
1943            **       C(A) --- number of elements in current stat array
1944            **       R2(S) -- same as A(S)
1945            **       R2(A) -- same as A(A)
1946            **       DO ----- same as A(A)
1947            **       P=0
1948            **       HEXMODE
1949            **       F-R0-0 - original DO if GETSDO
1950            **     otherwise: Fatal error
1951            **
1952            ** Calls:      ADRS50,B=DTOA,B=STAN
1953            **             GETSDO:  also SAVDO
1954            **
1955            ** Uses.......
1956            **  Inclusive: A,B,C(6-0),D(A),P,SB,R2,DO
1957            **             GETSDO: also F-R0-0
1958            **
1959            ** Stk lvls:   2
1960            **
1961            ** NOTE:
1962            **      Fatal error if there is no current statistical array,
1963            **      or if the current statistical array is invalid.
1964            **
1965            ** History:
1966            **
1967            **     Date     Programmer            Modification
1968            **   --------   ----------   ---------------------------------
1969            **   06/01/82      PM         Documented routine
1970            **   06/25/82       "         Replaced fatal errors with NaN's
1971            **   03/24/83       "         Replaced NaN's with fatal errors
1972            **
1973            ****************************************************************
1974            ****************************************************************
```

```
1975 0E54A 8F00  GETSD0 GOSBVL =SAVD0        save PC in F-R0-0
           000
1976 0E551 04   =GETSA  SETHEX
1977 0E553 7A70         GOSUB  B=STAN        read the current stat array name
1978 0E557 93E          ?C#0   X             any array selected?
1979 0E55A C0           GOYES  GETSA1
1980 0E55C 20    IVSTAR P=     0             error: no valid statistical array
1981 0E55E 3100         LC(2)  =eIVSAR         defined
1982 0E562 65CB         GOTO   mferr
1983 0E566 8E00  GETSA1 GOSUBL =ADRS50       get dope vector address
           00
1984 0E56C 4FE          GOC    IVSTAR
1985 0E56F 146          C=DAT0 A
1986 0E572 A26          C=C+C  XS            shift stat array bit out
1987 0E575 56E          GONC   IVSTAR        not a statistical array?
1988             *      B=C    XS            (stat arrays have base=0 now)
1989 0E578 80D3         P=C    3             P:=#VAR
1990 0E57C 165          D0=D0+ 6             get dimension bound
1991 0E57F 146          C=DAT0 A
1992 0E582 F6           CSR    A
1993             *      ?B#0   XS
1994             *      GOYES  GETSA3
1995 0E584 E6           C=C+1  A
1996 0E586 AFA   GETSA3 A=C    W             A(A)=length, A(S):=C(S)
1997 0E589 D2           C=0    A             get min length
1998 0E58B 80CF         C=P    15            C(S):=#VAR
1999 0E58F 809   GETSA4 C+P+1
2000 0E592 0D           P=P-1
2001 0E594 5AF          GONC   GETSA4        C(A)=min length
2002 0E597 ACE          ACEX   S             C(S) restored, A(S)=#VARS
2003 0E59A 8B2          ?C>A   A             min length>length?
2004 0E59D FB           GOYES  IVSTAR
2005 0E59F 102          R2=A                 R2(S):=#vars, R2(A):=length
2006 0E5A2 182          D0=D0- 3
2007 0E5A5 7130         GOSUB  B=DTOA         B(X):=dep#,indep#,#vars
2008 0E5A9 F6           CSR    A
2009 0E5AB 20           P=     0
2010 0E5AD 985          ?C>B   P             indep#>#vars?
2011 0E5B0 CA           GOYES  IVSTAR
2012 0E5B2 F6           CSR    A
2013 0E5B4 985          ?C>B   P             dep#>#vars?
2014 0E5B7 5A           GOYES  IVSTAR
2015 0E5B9 167          D0=D0+ 8
2016 0E5BC 146          C=DAT0 A
2017 0E5BF 132          ADOEX
2018 0E5C2 EE           C=A-C  A
2019 0E5C4 134          D0=C
2020 0E5C7 112          A=R2
2021 0E5CA DE           ACEX   A
2022 0E5CC 102          R2=A
2023 0E5CF 03           RTNCC
```

```
2024                    EJECT
2025                ****************************************************************
2026                ****************************************************************
2027                **
2028                ** Name:    B=STAN  -  Reads current statistical array name
2029                **
2030                ** Category:   LOCAL
2031                **
2032                ** Purpose:
2033                **      Reads current statistical array name.
2034                **
2035                ** Entry:
2036                **
2037                ** Exit:
2038                **      B(X) -- Current statistical array name
2039                **      C(X) -- same
2040                **      P      = 0
2041                **
2042                ** Calls:      nothing
2043                **
2044                ** Uses.......
2045                **  Inclusive: B(A),C(A),P,D0
2046                **
2047                ** Stk lvls:   0
2048                **
2049                ** History:
2050                **
2051                **    Date     Programmer              Modification
2052                **  --------   ----------   ----------------------------------
2053                **  01/06/83      PM         Revised documentation
2054                **
2055                ****************************************************************
2056                ****************************************************************
2057 0E5D1 20       B=STAN P=     0
2058 0E5D3 1B00           D0=(5) =STATAR
           000
2059 0E5DA 146      B=DTOA C=DATO A
2060 0E5DD D5               B=C    A
2061 0E5DF 01               RTN
```

```
2062                    EJECT
2063            ********************************************************************
2064            ********************************************************************
2065            **
2066            ** Name:    TBLU    -  Returns STAT element (L,K)
2067            ** Name:    ZTBLU   -  Returns STAT element (L,K) if K#0
2068            ** Name:    ZATBLU  -  Returns STAT element (K,K) if K#0
2069            ** Name:    TBLTOT  -  Returns STAT element (L,0)
2070            ** Name:    ST&GTN  -  Stacks A/B, Returns STAT element T(0,0)
2071            ** Name:       TBLU,ZTBLU,ZATBLU,TBLTOT,ST&GTN
2072            **
2073            ** Category:   MTHUTL
2074            **
2075            ** Purpose:
2076            **      The lower triangular part of the current statistical
2077            **      tableau is stored in the current statistical array by
2078            **      rows, with 16 nibs per statistical array element.
2079            **
2080            **      TBLU:    Returns element (L,K).
2081            **      ZTBLU:   If K#0, returns element (L,K).
2082            **               Otherwise, returns 0.
2083            **      ZATBLU:  If K#0, returns element (K,K).
2084            **               Otherwise, returns 0.
2085            **      TBLTOT:  Returns element (L,0).
2086            **      ST&GTN:  Pushes A/B onto top of math stack
2087            **               Returns element T(0,0)=N
2088            **
2089            **      In each case, a test for signaling NaN is made.
2090            **
2091            ** Entry:
2092            **      A(A) -- L (a valid hex variable number)
2093            **                (not used by ZATBLU)
2094            **      C(A) -- K (a valid hex variable number)
2095            **      R2(A) - points to first entry of the current statistical
2096            **              array
2097            **
2098            ** Exit:
2099            **      A/B --- T(L,K) split and normalized
2100            **      Carry=Set if signaling NaN
2101            **           =Clear otherwise
2102            **      DECMODE
2103            **      DO ---- points to element
2104            **
2105            ** Calls:      INVNaN,SPLITA
2106            **             ST&GTN also calls STSCR
2107            **             falls into SPLTAX
2108            **
2109            ** Uses.......
2110            **  Inclusive: A,B,C,P,DO,XM
2111            **
2112            ** Stk lvls:   2 for ST&GTN
2113            **             1 otherwise
2114            **
2115            ** History:
2116            **
```

```
2117                    **      Date      Programmer            Modification
2118                    **    --------    ----------    ---------------------------------
2119                    **    06/01/82       PM         Documented routine
2120                    **    12/16/82        "         Added signaling NaN test
2121                    **    01/06/83        "         Reviewed documentation
2122                    **
2123                    ********************************************************************
2124                    ********************************************************************
2125 0E5E1 111    ZTBLU+ A=R1
2126 0E5E4 118           C=R0
2127 0E5E7 8AE    ZTBLU  ?C#0    A
2128 0E5EA 71             GOYES  TBLU
2129 0E5EC DA     ZATBLU A=C     A
2130 0E5EE 8AE            ?C#0   A              if reg C has zero, return same
2131 0E5F1 01             GOYES  TBLU
2132 0E5F3 AF0            A=0    W              return zero in A/B
2133 0E5F6 543            GONC   SPLTAX         B.E.T.
2134 0E5F9 7F23   ST&GTN  GOSUB  STSCR          push A/B onto top of math scratch
2135 0E5FD D0             A=0    A
2136 0E5FF D2     TBLTOT C=0     A
2137 0E601 04    =TBLU   SETHEX
2138 0E603 8BA            ?A<=C  A
2139 0E606 40             GOYES  TBLU3
2140 0E608 DE             ACEX   A
2141 0E60A 80D0   TBLU3  P=C     0              here "L"=A(A)<=C(A)="K"
2142 0E60E D2             C=0    A
2143 0E610 0D     TBLU4  P=P-1
2144 0E612 480            GOC    TBLU5
2145 0E615 809            C+P+1
2146 0E618 57F            GONC   TBLU4          B.E.T.
2147 0E61B CA     TBLU5  A=A+C   A              A(A) has I=L+K(K+1)/2 in hex
2148 0E61D 11A            C=R2
2149 0E620 F0             ASL    A              16 nibs per statistics element
2150 0E622 C2             C=A+C  A
2151 0E624 134            DO=C
2152 0E627 1527           A=DAT0 W
```

```
2153                    EJECT
2154                ****************************************************************
2155                ****************************************************************
2156                **
2157                ** Name:(S) SPLTAX  -  Split, normalize A; handle signal NaN
2158                ** Name:(S) SIGTST  -  Handle signal NaN
2159                **
2160                ** Category:   MTHUTL
2161                **
2162                ** Purpose:
2163                **      SPLTAX:  Splits and normalizes contents of
2164                **                register A, then ...
2165                **      SIGTST:  Test for a signalling Nan and replace such a
2166                **               NaN by a quiet NaN.
2167                **
2168                ** Entry:
2169                **      SPLTAX:
2170                **        A --- argument in 12 digit form
2171                **      SIGTST:
2172                **        A/B = 15 digit form to be tested
2173                **
2174                ** Exit:
2175                **      A/B -- split and normalized argument
2176                **      Carry=Set:
2177                **             Signaling NaN replaced by a quiet NaN
2178                **             XM=1
2179                **      Carry=Clear, XM preserved, otherwise
2180                **
2181                ** Calls:      INVNaN,SPLITA
2182                **             may exit through invnan
2183                **
2184                ** Uses.......
2185                **  Inclusive: A,B,C(A),P,XM
2186                **
2187                ** Stk lvls:   1
2188                **
2189                ** Note:  Foreign NaNs are treated as signaling NaNs.
2190                **
2191                ** History:
2192                **
2193                **    Date     Programmer             Modification
2194                **  --------   ----------   ----------------------------------
2195                **  12/21/82      PM         Documented routine
2196                **  01/14/83       "         Revised documentation
2197                **
2198                ****************************************************************
2199                ****************************************************************
2200 0E62B 05   =SPLTAX SETDEC
2201 0E62D 8E00         GOSUBL =SPLITA        split and normalize into A/B
           00
2202 0E633 500          RTNNC                 return if finite (speed)
2203 0E636 20   =SIGTST P=     0
2204 0E638 3220         LCHEX  F02
           F
2205 0E63D 9BE          ?A>=C   X             signaling NaN? (works in decmode)
```

```
2206 0E640 40           GOYES  SIGNAN
2207 0E642 03           RTNCC                 no: return with carry clear
2208 0E644 3100  SIGNAN LC(2)  =eSIGOP        yes: create SIGOP NaN, set XM
2209 0E648 64EE         GOTO   invnan
```

```
2210                    EJECT
2211                ****************************************************************
2212                ****************************************************************
2213                **
2214                ** Name:    RANDOM  -  Randomize statement
2215                **
2216                ** Category:   STEXEC
2217                **
2218                ** Purpose:
2219                **      To construct a random number generator seed.
2220                **
2221                ** Entry:
2222                **      Optionally, a parsed seed numeric expression
2223                **
2224                ** Exit:
2225                **      15-digit seed at =RNSEED
2226                **
2227                ** Calls:      TEXPEX,POP1N+,hdflt
2228                **             exits through nxtstm
2229                **
2230                ** Uses.......
2231                **  Exclusive of EXPEX-: CPU: A,B,C,D,P,R3,D1,SB,XM,sIX(s7),
2232                **                               s8-11
2233                **                       RAM: RNSEED
2234                **
2235                ** Stk lvls:   max(4,1+EXPEXC) = max(2+uRES12,1+EXPEXC)
2236                **
2237                ** NOTE:
2238                **      The optional seed argument may be any real numeric
2239                **      expression.  A zero produces a "random" number
2240                **      sequence of zeroes.
2241                **
2242                ** History:
2243                **
2244                **     Date     Programmer            Modification
2245                **    -------   ----------   --------------------------------
2246                **    06/01/82      PM        Documented routine
2247                **    08/12/82       "        Removed complex type seed
2248                **    12/14/82       "        Added signaling NaN test
2249                **    01/26/83       "        Revised documentation
2250                **
2251                ****************************************************************
2252                ****************************************************************
2253 0E64C 0000         REL(5) =RDIZDC
          0
2254 0E651 0000         REL(5) =RANDP
          0
2255 0E656 77DB =RANDOM GOSUB  TEXPEX          get argument, if any
2256 0E65A 551          GONC   RNDMZD
2257 0E65D 1F00         D1=(5) =TIMER3         no arg: get clock time
          000
2258 0E664 143          A=DAT1 A               read least significant 5 nibs
2259 0E667 8E00         GOSUBL =hdflt          convert to decimal float-pt
          00
2260 0E66D 431          GOC    RNDMF           B.E.T.
```

```
2261 0E670 8E00  RNDMZD GOSUBL =POP1N+       pop argument, test for real
           00
2262 0E676 424          GOC    cnflc3         type, signaling NaN
2263 0E679 AC0          A=0    S
2264 0E67C 978          ?A=0   W
2265 0E67F 92           GOYES  PTSEED
2266 0E681 D1    RNDMF  B=0    A              nonzero:
2267 0E683 ABC          ABEX   X              put exponent into B(A)
2268 0E686 E5    RNDM2  B=B+1  A              increment exponent
2269 0E688 978          ?A=0   W              mantissa = 0?
2270 0E68B 51           GOYES  RNDM3
2271 0E68D D9           C=B    A              test for non-negative exponent
2272 0E68F A36          C=C+C  X
2273 0E692 5D0          GONC   RNDM3
2274 0E695 93A          ?C=0   X
2275 0E698 80           GOYES  RNDM3
2276 0E69A BF4          ASR    W              shift mantissa
2277 0E69D 58E          GONC   RNDM2          B.E.T.
2278 0E6A0 AB4   RNDM3  A=B    X              truncate last 3 mantissa digits
2279 0E6A3 BB0          ASL    X              append last 2 exponent digits, 1
2280 0E6A6 E4           A=A+1  A
2281 0E6A8 1F00  PTSEED D1=(5) =RNSEED        save seed X(0)
           000
2282 0E6AF 20           P=     0
2283 0E6B1 159E         DAT1=A 15
2284 0E6B5 6199         GOTO   nxtstm
2285            ****************************************************************
2286 0E6B9 6C42  cnflc3 GOTO   cnflct
```

```
2287                    EJECT
2288                ****************************************************************
2289                ****************************************************************
2290                **
2291                ** Name:    RND     -  Random Number Function
2292                **
2293                ** Category:   FNEXEC
2294                **
2295                ** Purpose:
2296                **      Computes the next pseudo-random number from the current
2297                **      seed value.
2298                **
2299                ** Entry:
2300                **      P=0
2301                **      seed value in RAM register
2302                **      Carry=Clear
2303                **
2304                ** Exit:
2305                **      number on top of math stack
2306                **      seed value changed for next RND call
2307                **
2308                ** Calls:      nothing
2309                **             exits through FNRTN1
2310                **
2311                ** Uses.......
2312                **  Inclusive: CPU: A,B,C,P
2313                **             RAM: RNSEED
2314                **
2315                ** Stk lvls:   1
2316                **
2317                ** Algorithm:
2318                **         Uses a linear congruential method
2319                **
2320                **                X(n):=C*X(n-1)  mod 10^15
2321                **
2322                **         to calculate the next seed value.
2323                **         RND is X(n)/10^15 truncated to 12 digits.
2324                **
2325                ** History:
2326                **
2327                **     Date      Programmer            Modification
2328                **    --------   ----------   ---------------------------------
2329                **    05/28/82      PM         Documented routine
2330                **    01/06/83       "         Revised documentation
2331                **
2332                ****************************************************************
2333                ****************************************************************
2334 0E6BD 00              NIBHEX 00
2335 0E6BF 05     =RND     SETDEC
2336 0E6C1 136             CDOEX                  save PC
2337 0E6C4 06              RSTK=C
2338 0E6C6 1B00            D0=(5) =RNSEED         load X(N) into B, 0 into A
          000
2339 0E6CD 15AE            A=DAT0 15              (P=0 here)
2340 0E6D1 AF1             B=0    W
```

```
2341 0E6D4 AFC          ABEX   W
2342 0E6D7 3F76         LCHEX  0002851130928467  load multiplier C
           4829
           0311
           5820
           00
2343 0E6E9 550          GONC   RND3             B.E.T.
2344 0E6EC A70   RND2   A=A+B  W
2345 0E6EF A0E   RND3   C=C-1  P
2346 0E6F2 59F          GONC   RND2
2347 0E6F5 BF1          BSL    W                shift X(N) left
2348 0E6F8 0C           P=P+1
2349 0E6FA 54F          GONC   RND3
2350 0E6FD 158E         DATO=A 15               save X(N+1)=C*X(N)
2351 0E701 AFC          ABEX   W                sign=exponent=0
2352 0E704 2E           P=     14
2353 0E706 919          ?B=0   WP               mantissa zero?
2354 0E709 11           GOYES  TRUNC            yes
2355 0E70B CC           A=A-1  A                exponent=-1
2356 0E70D 90D   RNDSHF ?B#0   P                normalized?
2357 0E710 A0           GOYES  TRUNC            yes
2358 0E712 CC           A=A-1  A                no: decrement exponent
2359 0E714 B91          BSL    WP                   left shift mantissa
2360 0E717 55F          GONC   RNDSHF           B.E.T.
2361 0E71A AD4   TRUNC  A=B    M                truncate last 3 mantissa digits
2362 0E71D 07           C=RSTK                  restore PC
2363 0E71F 134          D0=C
2364 0E722 AF6          C=A    W
2365 0E725 8C00         GOLONG =FNRTN1
           00
```

```
2366                    EJECT
2367            ****************************************************************
2368            ****************************************************************
2369            **
2370            ** Name:    FAC15S  -  Internal Factorial
2371            ** Name:    FACTF   -  Internal Factorial
2372            ** Name:(S) FCSTRT  -  Internal Factorial
2373            **
2374            ** Category:   MATH
2375            **
2376            ** Purpose:
2377            **      Computes the factorial of the 15-digit quantity in
2378            **      registers A/B.
2379            **
2380            ** Entry:
2381            **      A/B -- normalized 15-digit quantity
2382            **      user modes set
2383            **      DECMODE
2384            **
2385            ** Exit:
2386            **      A/B -- factorial in 15-digit form
2387            **      SB set if result is inexact
2388            **      XM set if NaN created
2389            **      Carry=Set
2390            **      DECMODE
2391            **
2392            ** Calls:      FNPWDS,INFR15,SHFMLT
2393            **             may exit through aornan
2394            **
2395            ** Uses.......
2396            **  Inclusive: A,B,C,D,P,SB,XM
2397            **
2398            ** Stk lvls:   2
2399            **
2400            ** NOTE:
2401            **      The result is accurate to 12 digits for all integer
2402            **      arguments i, where 0<=i<=253.  A noninteger finite
2403            **      or -Inf argument causes a NaN to be created and XM set.
2404            **
2405            ** Algorithm:
2406            **      A fast integer multiply method is used with adjustments
2407            **      for i=137 and 167 to insure full 12-digit accuracy.
2408            **
2409            ** History:
2410            **
2411            **     Date     Programmer            Modification
2412            **    --------   ----------   ---------------------------------
2413            **    05/28/82      PM         Documented routine
2414            **    06/25/82       "         Fatal errors for noninteger args
2415            **    01/06/83       "         Reviewed documentation
2416            **    01/13/83       "         NaN created for invalid args
2417            **
2418            ****************************************************************
2419            ****************************************************************
2420 0E72B 8E00 =FAC15S GOSUBL =FNPWDS        handle NaN, SB:=0
```

```
            00
2421 0E731 5B0          GONC   FACTF          jump if finite
2422 0E734 948          ?A=0   S              +inf?
2423 0E737 00           RTNYES
2424 0E739 638B  ARGOR  GOTO   aornan         -Inf: "eIVARG" NaN created, XM:=1
2425 0E73D 8E00  FACTF  GOSUBL =INFR15        finite: find decimal point
            00
2426 0E743 919          ?B=0   WP
2427 0E746 70           GOYES  FACT2
2428 0E748 88F          ?P#    15             noninteger?
2429 0E74B EE           GOYES  ARGOR
2430 0E74D 959   FACT2  ?B=0   M              treats -0 as 0
2431 0E750 70           GOYES  FCSTRT
2432 0E752 94C          ?A#0   S              <0?
2433 0E755 4E           GOYES  ARGOR          yes: fatal error
2434 0E757 AC0 =FCSTRT  A=0    S              use +0 for -0
2435 0E75A AC1          B=0    S
2436 0E75D D2           C=0    A              test for exponent in [0,2]
2437 0E75F E6           C=C+1  A
2438 0E761 E6           C=C+1  A
2439 0E763 8BA          ?A<=C  A
2440 0E766 31           GOYES  FACTOK
2441 0E768 AF0          A=0    W              exponent>2:
2442 0E76B B54          A=A+1  M                return large number (10^1000)
2443 0E76E AF1          B=0    W
2444 0E771 E5           B=B+1  A                inexact result: set SB
2445 0E773 F5           BSR    A
2446 0E775 6AA0         GOTO   FCM-1
2447 0E779 BF5   FACTOK BSR    W
2448 0E77C CC    FCALGN A=A-1  A              Align: right-justify integer in
2449 0E77E 480          GOC    FCXTST                 positions 15-13.
2450 0E781 BF1          BSL    W
2451 0E784 57F          GONC   FCALGN         B.E.T.
2452 0E787 822   FCXTST SB=0                  clear SB to detect inexactness
2453 0E78A AF3          D=0    W
2454 0E78D AF2          C=0    W              x=n, integer, aligned in reg B
2455 0E790 2E           P=     14
2456 0E792 B06          C=C+1  P              initial product = 1
2457 0E795 AFD          BCEX   W              C(15-13) has multiplier
2458                                          C(A) has exponent of part.prod.
2459                                          B    has mantissa of part.prod.
2460 0E798 949   FACMLT ?B=0   S              product didn't carry
2461 0E79B 21           GOYES  FACLSD
2462 0E79D E6           C=C+1  A              increment exponent
2463 0E79F 20           P=     0
2464 0E7A1 A05          B=B+B  P              see if round up
2465 0E7A4 BF5          BSR    W              shift mantissa
2466 0E7A7 550          GONC   FACLSD         skip if truncate
2467 0E7AA B75          B=B+1  W              round up (can't carry into S)
2468 0E7AD AF0   FACLSD A=0    W
2469 0E7B0 2D           P=     13
2470 0E7B2 B03          D=D-C  P
2471 0E7B5 5B0          GONC   FACISD         skip if LSD#0
2472 0E7B8 A70   FACLLP A=A+B  W              add partial product LSD times
2473 0E7BB B07          D=D+1  P
```

```
2474 0E7BE 59F          GONC   FACLLP
2475 0E7C1 2E    FACISD P=     14
2476 0E7C3 B03         D=D-C  P
2477 0E7C6 5C0         GONC   FACT9          skip if ISD=0
2478 0E7C9 BF3         DSL    W
2479 0E7CC 7550        GOSUB  SHFMLT         add 10*p.product ISD times
2480 0E7D0 4C0         GOC    FACMSD         B.E.T.
2481 0E7D3 94A   FACT9 ?C=0    S              enters here if ISD=0. MSD=0?
2482 0E7D6 11          GOYES  FACDEC         yes: finished with multiply
2483 0E7D8 BF4         ASR    W              no: shift extra time (truncate)
2484 0E7DB E6          C=C+1  A                  increment exponent
2485 0E7DD B43   FACMSD D=D-C  S
2486 0E7E0 560         GONC   FACDEC         skip if MSD=0
2487 0E7E3 7E30        GOSUB  SHFMLT         add 100*p.product MSD times
2488 0E7E7 AFC   FACDEC ABEX   W              Decrement multiplier:
2489 0E7EA 2D          P=     13
2490 0E7EC A0E         C=C-1  P                Decrement lsd. Carry?
2491 0E7EF 58A         GONC   FACMLT
2492 0E7F2 2E          P=     14
2493 0E7F4 A0E         C=C-1  P                Decrement isd. Carry?
2494 0E7F7 50A         GONC   FACMLT
2495 0E7FA A4E         C=C-1  S                Decrement msd. Carry?
2496 0E7FD 5A9         GONC   FACMLT
2497 0E800 AF1         B=0    W              done: get previous mantissa
2498 0E803 A9C         ABEX   WP
2499 0E806 DA          A=C    A
2500 0E808 20          P=     0              adjust mantissa for 167!
2501 0E80A 3400        LCHEX  00300
           300
2502 0E811 8A2         ?C=A   A              test exponent
2503 0E814 C0          GOYES  FCM-1
2504 0E816 3243        LCHEX  234            adjust mantissa for 137!
           2
2505 0E81B 8A6         ?C#A   A              test exponent
2506 0E81E 00          RTNYES
2507 0E820 A5D   FCM-1  B=B-1  M              decrement mantissa 1000 ulps
2508 0E823 02          RTNSC
```

```
2509                    EJECT
2510                ********************************************************************
2511                ********************************************************************
2512                **
2513                ** Name:    SHFMLT  -  Fast Multiply for Factorial
2514                **
2515                ** Category:   LOCAL
2516                **
2517                ** Purpose:
2518                **      Fast multiply loop for FACT function.  Increments
2519                **      exponent, shifts mantissa (rounding if necessary),
2520                **      and multiplies by next digit.
2521                **
2522                ** Entry:
2523                **      A ---- current product mantissa
2524                **      B ---- preceding (partial) product mantissa
2525                **      C(A) - current product exponent
2526                **      D(S) - 10 - multiplier digit (>0)
2527                **      DECMODE
2528                **
2529                ** Exit:
2530                **      A ---- new product mantissa
2531                **      B ---- preceding (partial) product mantissa
2532                **      C(A) - new product exponent
2533                **      P=0
2534                **      Carry=Set
2535                **      DECMODE
2536                **
2537                ** Calls:      nothing
2538                **
2539                ** Uses.......
2540                **  Inclusive: A,C(A),D(S),P
2541                **
2542                ** Stk lvls:   0
2543                **
2544                ** NOTE:
2545                **      See the FACT execution routine internal documentation
2546                **      for more information about how the registers are used.
2547                **
2548                ** History:
2549                **
2550                **    Date     Programmer             Modification
2551                **  --------   ----------   ---------------------------------
2552                **  06/23/82      PM         Documented routine
2553                **  01/06/83       "         Reviewed documentation
2554                **
2555                ********************************************************************
2556                ********************************************************************
2557 0E825 E6     SHFMLT C=C+1  A              increment exponent
2558 0E827 20            P=     0
2559 0E829 A04           A=A+A  P              see if round up
2560 0E82C BF4           ASR    W              shift mantissa
2561 0E82F 550           GONC   FSTMLT         skip if truncate
2562 0E832 B74           A=A+1  W              round up (can't carry into S)
2563 0E835 A70    FSTMLT A=A+B  W              fast multiply
```

```
2564 0E838 B47          D=D+1  S
2565 0E83B 59F          GONC   FSTMLT
2566 0E83E 01           RTN                  carry set on exit
```

```
2567                    EJECT
2568                ****************************************************************
2569                ****************************************************************
2570                **
2571                ** Name:    FACT    -  Factorial Function
2572                **
2573                ** Category:   FNEXEC
2574                **
2575                ** Purpose:
2576                **      FACT function execution routine.
2577                **
2578                ** Entry:
2579                **      numeric argument on top of math stack
2580                **      D1 points to top of math stack
2581                **
2582                ** Exit:
2583                **      result on top of math stack
2584                **
2585                ** Calls:      ARGPR+,FAC15S,ures12
2586                **             exits through outres
2587                **
2588                ** Uses.......
2589                **  Inclusive: A,B,C,D,P,R3,SB,XM,sIX(s7),s8-11
2590                **
2591                ** Stk lvls:   3 = max(3,1+uRES12)
2592                **
2593                ** History:
2594                **
2595                **    Date      Programmer              Modification
2596                **  --------    ----------   ------------------------------------
2597                **  06/23/82      PM         Documented routine
2598                **  01/26/83       "         Reviewed documentation
2599                **
2600                ****************************************************************
2601                ****************************************************************
2602 0E840 811            NIBHEX 811
2603 0E843 74A0 =FACT     GOSUB  ARGPR+          pop, test, prepare arg. SB,XM:=0
2604 0E847 70EE           GOSUB  =FAC15S
2605 0E84B 4C4            GOC    outrs2          B.E.T.
```

```
2606                    EJECT
2607                ****************************************************************
2608                ****************************************************************
2609                **
2610                ** Name:    FP      -  Fractional Part
2611                **
2612                ** Category:   FNEXEC
2613                **
2614                ** Purpose:
2615                **      FP function execution routine
2616                **
2617                ** Entry:
2618                **      numeric argument on top of math stack
2619                **      D1 points to top of math stack
2620                **
2621                ** Exit:
2622                **      result on top of math stack
2623                **
2624                ** Calls:      ARGPR+,FRAC15,ures12
2625                **             exits through outres
2626                **
2627                ** Uses.......
2628                **  Inclusive: A,B,C,D,P,R3,SB,XM,sIX(s7),s8-11
2629                **
2630                ** Stk lvls:   3 = max(3,1+uRES12)
2631                **
2632                ** History:
2633                **
2634                **    Date     Programmer             Modification
2635                **  --------   ----------   ----------------------------------
2636                **  06/23/82      PM         Documented routine
2637                **  01/26/83       "         Revised documentation
2638                **
2639                ****************************************************************
2640                ****************************************************************
2641 0E84E 811          NIBHEX 811
2642 0E851 7690 =FP    GOSUB  ARGPR+          pop, test, prepare arg. SB,XM:=0
2643 0E855 8E00        GOSUBL =FRAC15         get fractional part
           00
2644 0E85B 6C30        GOTO   outrs2
2645                ****************************************************************
2646                ****************************************************************
2647                **
2648                ** Name:    INT     -  Floor Function
2649                ** Name:    FLOOR   -  Floor Function
2650                **
2651                ** Category:   FNEXEC
2652                **
2653                ** Purpose:
2654                **      INT and FLOOR function execution routine
2655                **
2656                ** Entry:
2657                **      numeric argument on top of math stack
2658                **      D1 points to top of math stack
2659                **
```

```
2660            ** Exit:
2661            **      result on top of math stack
2662            **
2663            ** Calls:      SUBONE,argclf,ures12
2664            **             exits through outres
2665            **
2666            ** Uses.......
2667            **  Inclusive: A,B,C,D,P,R3,SB,XM,sIX(s7),s8-11
2668            **
2669            ** Stk lvls:   4 = max(4,1+uRES12)
2670            **
2671            ** History:
2672            **
2673            **    Date      Programmer            Modification
2674            **  --------    ----------   ----------------------------------
2675            **  06/23/82      PM         Documented routine
2676            **  01/26/83       "         Reviewed documentation
2677            **  03/04/83       "         Used SUBONE rather than subone
2678            **
2679            ********************************************************************
2680            ********************************************************************
2681 0E85F 811          NIBHEX 811
2682 0E862      =FLOOR
2683 0E862 7630 =INT    GOSUB  argclf         pop, test, prepare argument, setdec
2684                                           clear frac. part, preserve sign
2685 0E866 413          GOC    outrs2         done if result is exact (incl +-0)
2686 0E869 948          ?A=0   S              inexact positive number?
2687 0E86C C2           GOYES  outrs2         if so, done
2688 0E86E 8E00         GOSUBL =SUBONE        inexact negative number: decrement
         00
2689 0E874 6320         GOTO   outrs2           (SB=0 here)
2690            ********************************************************************
2691            ********************************************************************
2692            **
2693            ** Name:    CEIL    -  Ceiling Function
2694            **
2695            ** Category:   FNEXEC
2696            **
2697            ** Purpose:
2698            **      CEIL function execution routine
2699            **
2700            ** Entry:
2701            **      numeric argument on top of math stack
2702            **      D1 points to top of math stack
2703            **
2704            ** Exit:
2705            **      result on top of math stack
2706            **
2707            ** Calls:      ADDONE,argclf,ures12
2708            **             exits through outres
2709            **
2710            ** Uses.......
2711            **  Inclusive: A,B,C,D,P,R3,SB,XM,sIX(s7),s8-11
2712            **
2713            ** Stk lvls:   4 = max(4,1+uRES12)
```

```
2714                **
2715                ** History:
2716                **
2717                **    Date     Programmer            Modification
2718                **   --------  ----------  ----------------------------------
2719                **   06/23/82     PM        Documented routine
2720                **   01/26/83      "        Reviewed documentation
2721                **
2722                ********************************************************************
2723                ********************************************************************
2724 0E878 811            NIBHEX 811
2725 0E87B 7D10 =CEIL     GOSUB  argclf         pop, test, prepare argument
2726                                             clear frac. part, preserve sign
2727 0E87F 481            GOC    outrs2         done if result is exact (incl +-0)
2728 0E882 94C            ?A#0   S              inexact negative number?
2729 0E885 31             GOYES  outrs2         if so, done
2730 0E887 8E00           GOSUBL =ADDONE        inexact positive number: increment
          00
2731 0E88D 6A00           GOTO   outrs2           (SB=0 here)
2732                ********************************************************************
2733                ********************************************************************
2734                **
2735                ** Name:    IP      -  Integer Part Function
2736                **
2737                ** Category:   FNEXEC
2738                **
2739                ** Purpose:
2740                **      IP function execution routine
2741                **
2742                ** Entry:
2743                **      numeric argument on top of math stack
2744                **      D1 points to top of math stack
2745                **
2746                ** Exit:
2747                **      result on top of math stack
2748                **
2749                ** Calls:      argclf,ures12
2750                **             exits through outres
2751                **
2752                ** Uses.......
2753                **  Inclusive: A,B,C,D,P,R3,SB,XM,sIX(s7),s8-11
2754                **
2755                ** Stk lvls:   4 = max(4,1+uRES12)
2756                **
2757                ** History:
2758                **
2759                **    Date     Programmer            Modification
2760                **   --------  ----------  ----------------------------------
2761                **   06/23/82     PM        Documented routine
2762                **   01/26/83      "        Revised documentation
2763                **
2764                ********************************************************************
2765                ********************************************************************
2766 0E891 811            NIBHEX 811
2767 0E894 7400 =IP       GOSUB  argclf          pop, test, prepare argument
```

```
2768                                        clear fractional part
2769 0E898 604B  outrs2 GOTO   outres
2770            ************************************************************
2771            ************************************************************
2772            **
2773            ** Name:    argclf  -  Combine ARGPR+ and CLRFRC Routines
2774            **
2775            ** Category:   LOCAL
2776            **
2777            ** Purpose:
2778            **      Combines ARGPR+ and CLRFRC routines for packing.
2779            **
2780            ** Entry/Exit:
2781            **      See ARGPR+,CLRFRC
2782            **
2783            ** Calls:      ARGPR+
2784            **             exits through CLRFRC
2785            **
2786            ** Uses.......
2787            **  Inclusive: A,B,C(A),D(A),P,SB,XM,s8-11 unless fatal error
2788            **
2789            ** Stk lvls:   3
2790            **
2791            ************************************************************
2792            ************************************************************
2793 0E89C 7B40  argclf GOSUB  ARGPR+         pop, test, prepare arg. SB,XM:=0
2794 0E8A0 8C00         GOLONG =CLRFRC        clear fractional part
           00
```

```
2795                    EJECT
2796                ****************************************************************
2797                ****************************************************************
2798                **
2799                ** Name:    MAX     -  Maximum and Minimum Function
2800                ** Name:    MIN     -  Maximum and Minimum Function
2801                **
2802                ** Category:   FNEXEC
2803                **
2804                ** Purpose:
2805                **      Returns the maximum or minimum of 2 arguments.
2806                **      A NaN is created and XM is set if an unordered compare
2807                **      occurs (from a NaN arg.).
2808                **
2809                ** Entry:
2810                **      arguments are on the math stack
2811                **      HEXMODE
2812                **
2813                ** Exit:
2814                **      maximum or minimum argument or NaN is on the top
2815                **        of the math stack
2816                **      IVL flag set or cleared
2817                **      Fatal error for complex argument
2818                **
2819                ** Calls:      pop2n+,TST12A,XYEX,invnan,ures12
2820                **             exits through outres
2821                **
2822                ** Uses.......
2823                **  Inclusive: A,B,C,D,P,R3,D1,SB,XM,sIX(s7),s8-11
2824                **             unless fatal error
2825                **
2826                ** Stk lvls:   4 = 2+uRES12
2827                **
2828                ** Detail:
2829                **      This routine uses the <= or >= compare in TST12A to
2830                **      determine the maximum.  TST12A determines any
2831                **      unordered compare.
2832                **
2833                ** History:
2834                **
2835                **     Date     Programmer          Modification
2836                **   --------   ----------   --------------------------------
2837                **   05/26/82      PM        Documented routine
2838                **   11/16/82       "        Restricted argument count to 2
2839                **   11/30/82       "        Complex arg --> fatal error
2840                **   01/17/83       "        Signal any signaling NaN arg
2841                **   03/24/83       "        Revised documentation
2842                **
2843                ****************************************************************
2844                ****************************************************************
2845 0E8A6 8822           NIBHEX 8822
2846 0E8AA 7940 =MAX      GOSUB  pop2n+          pop args, test for signaling NaNs
2847 0E8AE 26             P=     6               >= test for MAX
2848 0E8B0 6A10           GOTO   MXENTR
2849 0E8B4 20    UNORC    P=     0               unordered: create NaN
```

```
2850 0E8B6 3100          LC(2)  =eUNORC          (XM,SB not needed here)
2851 0E8BA 7F6C          GOSUB  invnan
2852 0E8BE 49D           GOC    outrs2           B.E.T.
2853              ****************************************************************
2854 0E8C1 8822          NIBHEX 8822
2855 0E8C5 7E20 =MIN     GOSUB  pop2n+           pop args, test for signaling NaNs
2856 0E8C9 23            P=     3                <= test for MIN
2857 0E8CB 4A3   MXENTR  GOC    cnflct           complex?
2858 0E8CE 822           SB=0
2859 0E8D1 821           XM=0
2860 0E8D4 8E00          GOSUBL =TST12A
             00
2861 0E8DA 4DB           GOC    outrs2           tests true
2862 0E8DD 898           ?P=    8                unordered?
2863 0E8E0 4D            GOYES  UNORC
2864 0E8E2 8E00          GOSUBL =XYEX
             00
2865 0E8E8 5FA           GONC   outrs2           B.E.T.
```

```
2866                    EJECT
2867            ****************************************************************
2868            ****************************************************************
2869            **
2870            ** Name:(S) ARGPR+  -  Reads modes, pops and norm. real nbr
2871            **
2872            ** Category:   MTHSTK
2873            **
2874            ** Purpose:
2875            **      Reads user modes, pops numeric argument off math stack,
2876            **      tests for array or complex type or signaling NaN,
2877            **      splits and normalizes argument to 15-digit form,
2878            **      detects non-finiteness
2879            **
2880            ** Entry:
2881            **      Numeric argument on top of math stack
2882            **      D1 points to top of math stack
2883            **
2884            ** Exit:
2885            **      A/B -- 15-digit form of argument
2886            **      If signaling NaN: Carry=Set, XM=1
2887            **      Otherwise:        Carry=Clear
2888            **      DECMODE
2889            **      Fatal error if complex or array data type
2890            **
2891            ** Calls:      INVNaN,POP1R,SPLITA,umode+
2892            **
2893            ** Uses.......
2894            **  Inclusive: A,B,C(A),D(A),P,SB,XM,s8-11,
2895            **             unless fatal error
2896            **
2897            ** Stk lvls:   2
2898            **
2899            ** History:
2900            **
2901            **     Date     Programmer            Modification
2902            **   --------   ----------   --------------------------------
2903            **   05/26/82      PM         Documented routine
2904            **   12/14/82       "         Added signaling NaN test
2905            **   01/06/83       "         Revised documentation
2906            **
2907            ****************************************************************
2908            ****************************************************************
2909 0E8EB 7D29 =ARGPR+ GOSUB  umode+         get user modes, clear SB,XM
2910            ****************************************************************
2911            ****************************************************************
2912            **
2913            ** Name:(S) ARGPRP  -  Pops and normalizes real number
2914            **
2915            ** Category:   MTHSTK
2916            **
2917            ** Purpose:
2918            **      Same as ARGPR+, except that user modes are not read.
2919            **
2920            ** Entry:
```

```
2921            **      Same as ARGPR+
2922            **
2923            ** Exit:
2924            **      Same as ARGPR+, except user modes not read.
2925            **
2926            ** Calls:      INVNaN,POP1R,SPLITA
2927            **
2928            ** Uses.......
2929            **  Inclusive: A,B,C(A),P,XM, unless fatal error
2930            **
2931            ** Stk lvls:   2
2932            **
2933            ** History:
2934            **
2935            **     Date     Programmer            Modification
2936            **   --------   ----------   --------------------------------
2937            **   05/26/82      PM         Documented routine
2938            **   01/06/83       "         Revised documentation
2939            **
2940            ****************************************************************
2941            ****************************************************************
2942 0E8EF 7A00 =ARGPRP GOSUB  POP1R
2943 0E8F3 673D        GOTO   SPLTAX
2944            ****************************************************************
2945 0E8F7 8C00  pop2n+ GOLONG =POP2N+
         00
2946            ****************************************************************
2947            ****************************************************************
2948            **
2949            ** Name:(S) POP1R   -  Pops real number from math stack
2950            **
2951            ** Category:   MTHSTK
2952            **
2953            ** Purpose:
2954            **      pops numeric argument off the top of the math stack and
2955            **      tests that it is a real data type.
2956            **
2957            ** Entry:
2958            **      Numeric argument on top of math stack
2959            **      D1 points to top of math stack
2960            **
2961            ** Exit:
2962            **      A -- has 12-digit form of argument
2963            **      Carry=clear
2964            **      DECMODE
2965            **      fatal error if array or complex data type
2966            **
2967            ** Calls:      POP1N
2968            **
2969            ** Uses.......
2970            **  Inclusive: A,B(X),P, unless fatal error
2971            **
2972            ** Stk lvls:   1
2973            **
2974            ** History:
```

```
2975                ** 
2976                **    Date     Programmer             Modification
2977                ** --------   ----------   ---------------------------------
2978                ** 08/12/82      PM         Documented routine
2979                ** 01/06/83       "         Revised documentation
2980                ** 
2981                ********************************************************************
2982                ********************************************************************
2983 0E8FD 8E00 =POP1R  GOSUBL =POP1N
           00
2984 0E903 500          RTNNC
2985 0E906 8C00  cnflct GOLONG =CNFLCT        complex arg: fatal error
           00
2986                ********************************************************************
2987                ********************************************************************
2988                ** 
2989                ** Name:(S) ARGSTA  -  Pops and tests real number
2990                ** Name:(S) ARGST-  -  Pops and tests real number
2991                ** 
2992                ** Category:   MTHSTK
2993                ** 
2994                ** Purpose:
2995                **      Reads user modes, pops numeric argument off math stack,
2996                **      tests for array or complex type, detects non-
2997                **      finiteness, and tests for NaN.
2998                ** 
2999                ** Entry:
3000                **      Numeric argument on top of math stack
3001                ** 
3002                ** Exit:
3003                **      A ---- 12-digit argument from top of stack
3004                **      Carry=Clear if real finite
3005                **      Carry=Set if infinity
3006                **      Fatal error if array, complex, or NaN
3007                **      DECMODE
3008                ** 
3009                ** Calls:      POP1R,finita,umode+
3010                ** 
3011                ** Uses.......
3012                **  Inclusive: A,B(X),D(A),P
3013                **             ARGSTA: also SB,XM,s8-11
3014                ** 
3015                ** Stk lvls:   2
3016                ** 
3017                ** NOTE:
3018                **      Input      Fatal error message
3019                ** 
3020                **      array        "eDATTY"
3021                **      complex      "eDATTY"
3022                **      NaN          "eIVARG"
3023                ** 
3024                ** History:
3025                ** 
3026                **    Date     Programmer             Modification
3027                ** --------   ----------   ---------------------------------
```

```
3028            ** 07/16/82      PM         Documented routine
3029            ** 10/06/82       "         Removed projective infinity test
3030            ** 01/06/83       "         Revised documentation
3031            **
3032            ********************************************************************
3033            ********************************************************************
3034 0E90C 7C09 =ARGSTA GOSUB  umode+         read user modes
3035 0E910 79EF =ARGST- GOSUB  POP1R          pop off math stack
3036 0E914 7E00         GOSUB  finita         test for nonfinite
3037 0E918 500          RTNNC                 return if finite with carry clear
3038 0E91B 968          ?A=0   B              Inf?
3039 0E91E 00           RTNYES                  yes: return with carry set
3040            ********************************************************************
3041            ********************************************************************
3042            **
3043            ** Name:(S) IVAERR  -  Report "Invalid Arg" error.
3044            **
3045            ** Category:   SYSTEM
3046            **
3047            ** Purpose:
3048            **      To do a GOLONG =ARGERR
3049            **
3050            ** History:
3051            **
3052            **    Date     Programmer              Modification
3053            **  --------   ----------   ------------------------------------
3054            **  11/09/83   MB           Documentation
3055            **
3056            ********************************************************************
3057            ********************************************************************
3058 0E920 8C00 =IVAERR GOLONG =ARGERR        invalid argument error
          00
3059            ********************************************************************
3060 0E926 8C00  finita GOLONG =FINITA
          00
```

```
3061                    EJECT
3062              ****************************************************************
3063              ****************************************************************
3064              **
3065              ** Name:(S) STSCR   -  Push 15-Form Onto Math Scratch Stack
3066              **
3067              ** Category:   MTHUTL
3068              **
3069              ** Purpose:
3070              **      Pushes a 15-digit form onto top of math scratch stack
3071              **
3072              ** Entry:
3073              **      A(S) ---- sign
3074              **      A(A) ---- exponent
3075              **      B(14-0) - mantissa
3076              **
3077              ** Exit:
3078              **      P = 1
3079              **      Carry=Clear
3080              **
3081              ** Calls:      GEXPAD,GSCPTR
3082              **
3083              ** Uses.......
3084              **  Inclusive: C,D0,P
3085              **
3086              ** Stk lvls:   1
3087              **
3088              ** History:
3089              **
3090              **    Date      Programmer              Modification
3091              **  --------    ----------   ---------------------------------
3092              **  ??/??/82       BS         Wrote and coded routines
3093              **  12/07/82       PM         Packed and documented routines
3094              **  01/06/83        "         Reviewed documentation
3095              **
3096              ****************************************************************
3097              ****************************************************************
3098 0E92C 7A90 =STSCR  GOSUB  GSCPTR          Get address of top of scratch stack
3099 0E930 B06         C=C+1  P                Increment stack pointer
3100 0E933 0B          CSTEX                   Swap into status and
3101 0E935 846         ST=0   6                  clear upper 2 bits of second nib
3102              *    ST=0   7                  (Mod 4 reduction)
3103 0E938 0B          CSTEX                   Then swap back into register
3104 0E93A 1540        DAT0=C P                Write out updated stack pointer nib
3105 0E93E 134         D0=C                    Put address in data pointer
3106 0E941 AF9         C=B    W                Copy mantissa from B
3107 0E944 AC6         C=A    S                Copy sign from A
3108 0E947 1547        DAT0=C W                Write out sign and mantissa
3109 0E94B 7390        GOSUB  GEXPAD           Get address of exponent
3110 0E94F 140         DAT0=A A                Write out exponent
3111 0E952 03          RTNCC                   Done
```

```
3112                    EJECT
3113                ****************************************************************
3114                ****************************************************************
3115                **
3116                ** Name:(S) RCSCR   -  Pop 15-Form From Math Scratch Stack
3117                **
3118                ** Category:   MTHUTL
3119                **
3120                ** Purpose:
3121                **      Pops a 15-digit form from scratch stack
3122                **
3123                ** Entry:
3124                **
3125                ** Exit:
3126                **      C(S) ---- sign
3127                **      C(A) ---- exponent
3128                **      D(14-0) - mantissa
3129                **      A/B ----- unchanged
3130                **      Carry=Clear
3131                **      P = 1
3132                **
3133                ** Calls:      GEXPAD,GSCPTR
3134                **
3135                ** Uses.......
3136                **  Inclusive: C,D,D0,P
3137                **
3138                ** Stk lvls:   1
3139                **
3140                ** History:
3141                **
3142                **    Date      Programmer              Modification
3143                **  --------    ----------   ----------------------------------
3144                **  ??/??/82       BS         Wrote and coded routines
3145                **  12/07/82       PM         Packed and documented routines
3146                **  01/06/83        "         Reviewed documentation
3147                **
3148                ****************************************************************
3149                ****************************************************************
3150 0E954 7270 =RCSCR  GOSUB  GSCPTR          Get address of top of scratch stack
3151 0E958 A87         D=C    P               ****BUG FIXED -- PM****
3152 0E95B A0E         C=C-1  P               Decrement stack pointer
3153 0E95E 550         GONC   RCSCR1          Did it borrow from 0 to F?
3154 0E961 303         LCHEX  3               Yes, then set it to 3
3155 0E964 1540  RCSCR1 DAT0=C P              Write out new stack pointer
3156 0E968 A8B         C=D    P               ****BUG FIXED -- PM****
3157 0E96B 134         D0=C                   Copy into data pointer
3158 0E96E 1567        C=DAT0 W               Read in mantissa and sign
3159 0E972 AF7         D=C    W               Move mantissa to D
3160 0E975 7960        GOSUB  GEXPAD          Get exponent address
3161 0E979 146         C=DAT0 A               Read in exponent
3162 0E97C AC3         D=0    S               Clear sign in D
3163 0E97F 03          RTNCC                  Done
```

```
3164                    EJECT
3165            ****************************************************************
3166            ****************************************************************
3167            **
3168            ** Name:    EXSCR   -  Exchange 15-Form w/Math Scrtch Stack Top
3169            **
3170            ** Category:   MTHSTK
3171            **
3172            ** Purpose:
3173            **      Exchange 15-digit form with top of math scratch stack
3174            **
3175            ** Entry:
3176            **      A(S) ---- sign
3177            **      A(A) ---- exponent
3178            **      B(14-0) - mantissa
3179            **
3180            ** Exit:
3181            **      A/B exchanged with top of math scratch stack
3182            **      C/D unchanged
3183            **      P=1
3184            **      Carry=Clear
3185            **
3186            ** Calls:      GSCPTR
3187            **
3188            ** Uses.......
3189            **  Inclusive: A,B,D0,P
3190            **
3191            ** Stk lvls:   2
3192            **
3193            ****************************************************************
3194            ****************************************************************
3195            *EXSCR  B=A    S              Copy sign into B
3196            *       RSTK=C                Save C(A){exponent} on stack
3197            *       GOSUB  GSCPTR         Get address of top of scratch stack
3198            *       CSTEX                 Swap address into status
3199            *       ST=1   6                and select exponent fields
3200            *       CSTEX                 Then move address into D0
3201            *       D0=C
3202            *       C=DAT0 A              Read exponent
3203            *       DAT0=A A              Exchange exponents
3204            *       CDOXS                 Swap address into status
3205            *       CSTEX
3206            *       ST=0   6                and select mantissa fields
3207            *       CSTEX                 Then swap back into data pointer
3208            *       CDOXS                 11/22/82 fix by PM
3209            *       A=DAT0 W              Read mantissa
3210            *       ABEX   W
3211            *       DAT0=A W              Exchange mantissas
3212            *       A=C    A              Copy exponent into A
3213            *       C=RSTK                Recall C(A){exponent}
3214            *       A=B    S              Copy sign into A
3215            *       B=0    S              Clear sign of B
3216            *       RTNCC                 Done
```

```
3217                    EJECT
3218                ****************************************************************
3219                ****************************************************************
3220                **
3221                ** Name:(S) RCLW1   -  Recall 1st (Top) Math Scrtch Stack Entry
3222                ** Name:(S) RCLW2   -  Recall 2nd Math Scratch Stack Entry
3223                ** Name:(S) RCLW3   -  Recall 3rd Math Scratch Stack Entry
3224                ** Name:    RCLW4   -  Recall 4th Math Scratch Stack Entry
3225                ** Name:(S) RCL*    -  Recall Selected Math Scratch Stack Entry
3226                **
3227                ** Category:   MTHUTL
3228                **
3229                ** Purpose:
3230                **      Move the 15-digit form in A/B to C/D and then recall
3231                **      the requested math scratch stack entry in A/B without
3232                **      removing that entry from the stack.
3233                **
3234                ** Entry:
3235                **      (A,B)  =  15-form number
3236                **     RCL*:
3237                **      P      =  0 for 1st entry on math scratch stack
3238                **             =  n-1 for nth entry on math scratch stack
3239                **
3240                ** Exit:
3241                **      (A,B)  =  15-form number from match scratch stack
3242                **      (C,D)  =  (A,B) on entry
3243                **      P      =  1
3244                **      DECMODE
3245                **      Carry  =  Clear
3246                **
3247                ** Calls:      GEXPAD,GSCPTR
3248                **
3249                ** Uses.......
3250                **  Inclusive: A,B,C,D,D0,P
3251                **
3252                ** Stk lvls:   1
3253                **
3254                ** History:
3255                **
3256                **    Date      Programmer             Modification
3257                **  --------    ----------   ----------------------------------
3258                **  ??/??/82       BS         Wrote and coded routines
3259                **  12/07/82       PM         Packed and documented routines
3260                **  01/06/83       PM         Reviewed documentation
3261                **
3262                ****************************************************************
3263                ****************************************************************
3264 0E981 20      =RCLW1  P=     0               Specify recall stack level 1
3265 0E983 04      =RCL*   SETHEX
3266 0E985 80F1            CPEX   1               Swap pointer into second nib of add
3267 0E989 AE7             D=C    B               Copy pointer to D
3268 0E98C 7A30            GOSUB  GSCPTR          Get address of top of scratch stack
3269 0E990 B0B             C=C-D  P               Offset by stack register number
3270 0E993 0B              CSTEX                  Swap into status
3271 0E995 846             ST=0   6                 to clear upper 2 bits of nibble 2
```

```
3272 0E998 847          ST=0   7            of address space (mod 4 reduct.)
3273 0E99B 0B           CSTEX               Then swap back into register
3274 0E99D 134          DO=C                Point to selected register
3275 0E9A0 1567         C=DATO W            Read desired register
3276 0E9A4 AFD          BCEX   W            Swap recalled mantissa into B
3277 0E9A7 AF7          D=C    W            Move old B register to D
3278 0E9AA 7430         GOSUB  GEXPAD       Get address of exponent
3279 0E9AE 146          C=DATO A            Now read exponent for desired reg.
3280 0E9B1 AFE          ACEX   W            Swap recalled exponent into A
3281 0E9B4 05           SETDEC
3282 0E9B6 AC4          A=B    S            Copy sign nibble from B
3283 0E9B9 AC1          B=0    S            Clear sign in B
3284 0E9BC 03           RTNCC               Done
3285 0E9BE 21   =RCLW2  P=     1            Specify recall stack level 2
3286 0E9C0 62CF         GOTO   RCL*         Join recall code
3287 0E9C4 22   =RCLW3  P=     2            Specify recall stack level 3
3288 0E9C6 6CBF         GOTO   RCL*         Join recall code
3289            *RCLW4  P=     3            Specify recall stack level 4
3290            *       GOTO   RCL*
```

```
3291                    EJECT
3292                ********************************************************************
3293                ********************************************************************
3294                **
3295                ** Name:    GSCPTR  -  Scratch math stack utility
3296                **
3297                ** Category:   LOCAL
3298                **
3299                ** Purpose:
3300                **      Gets address of top of scratch math stack.
3301                **
3302                ** Entry:
3303                **      nothing in particular
3304                **
3305                ** Exit:
3306                **      C(A) -- points to top of scratch math stack
3307                **
3308                ** Calls:      nothing
3309                **
3310                ** Uses.......
3311                **  Inclusive: C(A),P,D0
3312                **
3313                ** Stk lvls:   0
3314                **
3315                ** History:
3316                **
3317                **    Date     Programmer              Modification
3318                **  --------   ----------   ----------------------------------
3319                **  01/06/83      PM         Documented routine
3320                **
3321                ********************************************************************
3322                ********************************************************************
3323 0E9CA 1B00  GSCPTR D0=(5) =SCRPTR        Get address of top of scratch stack
           000
3324 0E9D1 20           P=     0
3325 0E9D3 3400         LC(5)  =SCRSTO          Pointer to stack register 1
           000
3326 0E9DA 21           P=     1                Select second nibble of address
3327 0E9DC 1560         C=DAT0 P                Recall current stack pointer
3328 0E9E0 01           RTN
```

```
3329                    EJECT
3330                ****************************************************************
3331                ****************************************************************
3332                **
3333                ** Name:    GEXPAD  -  Scratch math stack utility
3334                **
3335                ** Category:   LOCAL
3336                **
3337                ** Purpose:
3338                **      Given address of scratch math stack element's sign and
3339                **      mantissa, finds address of its exponent.
3340                **
3341                ** Entry:
3342                **      D0 --- adress of sign/mantissa
3343                **
3344                ** Exit:
3345                **      D0 --- address of exponent
3346                **
3347                ** Calls:      nothing
3348                **
3349                ** Uses.......
3350                **  Inclusive: C(3-0),D0
3351                **
3352                ** Stk lvls:   0
3353                **
3354                ** History:
3355                **
3356                **    Date     Programmer             Modification
3357                **  --------   ----------   ---------------------------------
3358                **  01/06/83      PM         Documented routine
3359                **
3360                ****************************************************************
3361                ****************************************************************
3362 0E9E2 13E    GEXPAD CDOXS                  Get address of exponent:
3363 0E9E5 0B          CSTEX                     Move address into status
3364 0E9E7 856         ST=1   6                    to select expontent's address
3365 0E9EA 0B          CSTEX                     then swap back
3366 0E9EC 13C         D0=CS                     and move back to data pointer
3367 0E9EF 01          RTN
3368 0E9F1             END
```

```
 ABORTS  Abs   57065 #0DEE9 -   176   185   189
=AD15S   Abs   57757 #0E19D -   739   167   698   871  1422
 AD15s   Ext                 -   740
=ADD     Abs   56946 #0DE72 -   138
 ADDDC   Ext                 -   136
 ADDEND  Abs   57402 #0E03A -   291   272
 ADDONE  Ext                 -  2730
 ADDP    Ext                 -   137
 ADJSQ   Abs   56969 #0DE89 -   147
 ADJSQ2  Abs   56980 #0DE94 -   150   174
 ADJSQ3  Abs   56989 #0DE9D -   153   155
 ADJSQ4  Abs   57073 #0DEF1 -   179   151
 ADJSQ6  Abs   57179 #0DF5B -   220   199   260
 ADJSQ7  Abs   57207 #0DF77 -   228   244
 ADJSQ8  Abs   57258 #0DFAA -   245   233
 ADJSQN  Abs   57042 #0DED2 -   168   160
 ADJTOT  Abs   57309 #0DFDD -   261   222
 ADRS50  Ext                 -  1591  1983
 ARESD1  Abs   57834 #0E1EA -   871   286
 ARGERR  Ext                 -  3058
 ARGOR   Abs   59193 #0E739 -  2424  2429  2433
=ARGPR+  Abs   59627 #0E8EB -  2909  2603  2642  2793
=ARGPRP  Abs   59631 #0E8EF -  2942  1416
=ARGST-  Abs   59664 #0E910 -  3035
=ARGSTA  Abs   59660 #0E90C -  3034
 ATOT0   Abs   57185 #0DF61 -   222   208
 ATOT1   Abs   57333 #0DFF5 -   271   290
 ATOT2   Abs   57342 #0DFFE -   274   276
 ATOT3   Abs   57385 #0E029 -   286   284
 AVMEME  Ext                 -   928
 B=DTOA  Abs   58842 #0E5DA -  2059  2007
 B=STAN  Abs   58833 #0E5D1 -  2057  1590  1977
 BFGTAB  Abs   57680 #0E150 -   558   234   249
 BFGTCD  Abs   57696 #0E160 -   601   238   252
 BFPTAB  Abs   57712 #0E170 -   647   168   253
=CEIL    Abs   59515 #0E87B -  2725
 CLRFRC  Ext                 -  2794
=CLSTAT  Abs   56910 #0DE4E -    73
 CLSTLP  Abs   56917 #0DE55 -    75    79
 CNFLCT  Ext                 -  2985
 COLLAP  Ext                 -  1017
=CORR    Abs   58203 #0E35B -  1475
 CORR2   Abs   58319 #0E3CF -  1513  1511
 CORR3   Abs   58323 #0E3D3 -  1515  1296  1360  1424  1483  1486  1489
 Cslc5   Ext                 -   974
 D0=SR0  Abs   57423 #0E04F -   300   148   194   473
 D1+21   Abs   57454 #0E06E -   312   170   243   257   282   463
 D1-16   Abs   57730 #0E182 -   653   562   605
 D1=AVE  Ext                 -  1645
 D1=SR1  Abs   57439 #0E05F -   307   147   196   437   472
 D1STOR  Ext                 -  1651
 DCHXF   Ext                 -  1216
 DENTRY  Abs   56953 #0DE79 -   140    30
 DEST    Ext                 -  1639
=DROP    Abs   56889 #0DE39 -    28
```

```
 DROPDC  Ext                  -    26
 DROPP   Ext                  -    27
 DV15S   Ext                  -  1806
 EXPEXC  Ext                  -  1024
=FAC15S  Abs   59179 #0E72B -  2420  2604
 FACDEC  Abs   59367 #0E7E7 -  2488  2482  2486
 FACISD  Abs   59329 #0E7C1 -  2475  2471
 FACLLP  Abs   59320 #0E7B8 -  2472  2474
 FACLSD  Abs   59309 #0E7AD -  2468  2461  2466
 FACMLT  Abs   59288 #0E798 -  2460  2491  2494  2496
 FACMSD  Abs   59357 #0E7DD -  2485  2480
=FACT    Abs   59459 #0E843 -  2603
 FACT2   Abs   59213 #0E74D -  2430  2427
 FACT9   Abs   59347 #0E7D3 -  2481  2477
 FACTF   Abs   59197 #0E73D -  2425  2421
 FACTOK  Abs   59257 #0E779 -  2447  2440
 FBWVAR  Abs   57528 #0E0B8 -   435   262
 FBWVR1  Abs   57568 #0E0E0 -   446   465
 FBWVR2  Abs   57611 #0E10B -   460   449
 FBWVR3  Abs   57617 #0E111 -   462   458
 FBWVR4  Abs   57631 #0E11F -   466   447
 FBWVR5  Abs   57648 #0E130 -   472   467
 FCALGN  Abs   59260 #0E77C -  2448  2451
 FCM-1   Abs   59424 #0E820 -  2507  2446  2503
=FCSTRT  Abs   59223 #0E757 -  2434  2431
 FCXTST  Abs   59271 #0E787 -  2452  2449
 FINITA  Ext                  -  3060
=FLOOR   Abs   59490 #0E862 -  2682
 FNPWDS  Ext                  -  2420
 FNRTN1  Ext                  -  2365
 FNRTN4  Ext                  -  1517
=FP      Abs   59473 #0E851 -  2642
 FRAC15  Ext                  -  2643
 FSTMLT  Abs   59445 #0E835 -  2563  2561  2565
 GETMSG  Abs   57896 #0E228 -   973   176   813
=GETSA   Abs   58705 #0E551 -  1976    73   140  1582
 GETSA1  Abs   58726 #0E566 -  1983  1979
 GETSA3  Abs   58758 #0E586 -  1996
 GETSA4  Abs   58767 #0E58F -  1999  2001
 GETSDO  Abs   58698 #0E54A -  1975  1156  1404  1475
 GEXPAD  Abs   59874 #0E9E2 -  3362  3109  3160  3278
 GSCPTR  Abs   59850 #0E9CA -  3323  3098  3150  3268
 GTINSL  Abs   58553 #0E4B9 -  1701  1413  1599
 I/OALL  Ext                  -   379
 I/ODAL  Ext                  -   293
 INFR15  Ext                  -  2425
=INT     Abs   59490 #0E862 -  2683
 INVNaN  Ext                  -  1909
=IP      Abs   59540 #0E894 -  2767
=IVAERR  Abs   59680 #0E920 -  3058  1573
 IVARG   Ext                  -  1230
 IVP     Ext                  -   812
 IVSOPR  Abs   58677 #0E535 -  1912  1852
 IVSTAR  Abs   58716 #0E55C -  1980  1984  1987  2004  2011  2014
 IVSTAT  Abs   58663 #0E527 -  1873  1918
```

```
 IVVRNB  Abs   58034 #0E2B2 -  1226  1215
=LR      Abs   58364 #0E3FC -  1579
 LR1     Abs   58455 #0E457 -  1608  1576
 LRDC    Ext                -  1577
 LREND   Abs   58549 #0E4B5 -  1658  1632
 LRIVAE  Abs   58339 #0E3E3 -  1573  1584  1588
 LRP     Ext                -  1578
 LRSKIP  Abs   58343 #0E3E7 -  1574  1600
 LRVRLP  Abs   58480 #0E470 -  1622  1657
=MAX     Abs   59562 #0E8AA -  2846
=MEAN    Abs   58065 #0E2D1 -  1284
 MEAN1   Abs   58068 #0E2D4 -  1285  1282
 MEAN3   Abs   58088 #0E2E8 -  1296  1289  1293
 MFERR   Ext                -   471
=MFERRO  Abs   57640 #0E128 -   469
=MIN     Abs   59589 #0E8C5 -  2855
 MP15S   Ext                -   765
 MSARES  Abs   57816 #0E1D8 -   845   239   254
 MSN12   Ext                -  1482
 MTHSTK  Ext                -   440
 MXENTR  Abs   59595 #0E8CB -  2857  2848
 NNGFIN  Abs   58686 #0E53E -  1915  1868
 NNGTAB  Abs   58646 #0E516 -  1867   188  1350  1850
 NOMEM   Ext                -  1103
 NXTSTM  Ext                -   295
 ORSB    Ext                -   741
 POP1N   Ext                -  2983
 POP1N+  Ext                -  2261
=POP1R   Abs   59645 #0E8FD -  2983  1213  2942  3035
 POP2N+  Ext                -  2945
 POPMTH  Ext                -  1646
 POSTAB  Abs   58636 #0E50C -  1849   184  1354  1498  1504  1803
=PREDV   Abs   58141 #0E31D -  1404
 PREDV1  Abs   58191 #0E34F -  1424
 PRTNCC  Abs   58696 #0E548 -  1919  1870  1872  1916
 PTRCDV  Abs   58618 #0E4FA -  1803  1357  1708
 PTSEED  Abs   59048 #0E6A8 -  2281  2265
=RANDOM  Abs   58966 #0E656 -  2255
 RANDP   Ext                -  2254
 RCCD1X  Ext                -   165   281   849
=RCL*    Abs   59779 #0E983 -  3265  3286  3288
=RCLW1   Abs   59777 #0E981 -  3264  1063  1421  1509  1714  1805
=RCLW2   Abs   59838 #0E9BE -  3285  1605
=RCLW3   Abs   59844 #0E9C4 -  3287  1418
=RCSCR   Abs   59732 #0E954 -  3150  1506  1721
 RCSCR1  Abs   59748 #0E964 -  3155  3153
 RDIZDC  Ext                -  2253
=RND     Abs   59071 #0E6BF -  2335
 RND2    Abs   59116 #0E6EC -  2344  2346
 RND3    Abs   59119 #0E6EF -  2345  2343  2349
 RNDM2   Abs   59014 #0E686 -  2268  2277
 RNDM3   Abs   59040 #0E6A0 -  2278  2270  2273  2275
 RNDMF   Abs   59009 #0E681 -  2266  2260
 RNDMZD  Abs   58992 #0E670 -  2261  2256
 RNDSHF  Abs   59149 #0E70D -  2356  2360
```

```
 RNSEED  Ext                    -  2281  2338
=RPTNaN  Abs   57808 #0E1D0     -   817   169
 RSTDO   Ext                    -  1515
 RTNCC   Ext                    -    72
 S-R0-0  Ext                    -   300   368
 S-R0-1  Ext                    -   224   307   383
 S-R0-3  Ext                    -   438
 SAVDO   Ext                    -  1975
 SAVESB  Ext                    -   739
=SB15S   Abs   57754 #0E19A     -   738  1722
 SCRPTR  Ext                    -  3323
 SCRSTO  Ext                    -  3325
=SDEV    Abs   58095 #0E2EF     -  1342
 SDEV2   Abs   58134 #0E316     -  1360  1346
 SGNPRD  Abs   58614 #0E4F6     -  1763  1294
 SHFMLT  Abs   59429 #0E825     -  2557  2479  2487
 SIGNAN  Abs   58948 #0E644     -  2208  2206
=SIGTST  Abs   58934 #0E636     -  2203
 SPLITA  Ext                    -  2201
=SPLTAS  Abs   57663 #0E13F     -   515   157   280   519
=SPLTAX  Abs   58923 #0E62B     -  2200   515  1228  2133  2943
 SQADD2  Abs   57104 #0DF10     -   188   183
 SQRSAV  Ext                    -  1359  1508
 ST&GTN  Abs   58873 #0E5F9     -  2134  1352  1763
 STAB1   Ext                    -   278   847
 STAB1X  Ext                    -   163
 STATAR  Ext                    -  2058
 STBAL1  Abs   57514 #0E0AA     -   382   380
 STBALL  Abs   57462 #0E076     -   367   141
 STFPRP  Abs   57968 #0E270     -  1156  1285  1342
 STK16?  Ext                    -  1100
 STMTD1  Ext                    -   926   932
 STOPDC  Ext                    -    71
=STSCR   Abs   59692 #0E92C     -  3098  1026  1415  1494  1500  1704  1710  1717
                                   2134
 SUBONE  Ext                    -  2688
 SVTRC   Ext                    -  1637
 SWAPXY  Ext                    -   186   248
 TBLTOT  Abs   58879 #0E5FF     -  2136  1290  1712  1719
=TBLU    Abs   58881 #0E601     -  2137  1492  2128  2131
 TBLU3   Abs   58890 #0E60A     -  2141  2139
 TBLU4   Abs   58896 #0E610     -  2143  2146
 TBLU5   Abs   58907 #0E61B     -  2147  2144
 TEXPEX  Abs   57905 #0E231     -  1017    28   138  1579  2255
 TIMER3  Ext                    -  2257
=TOTAL   Abs   58056 #0E2C8     -  1281
 TRUNC   Abs   59162 #0E71A     -  2361  2354  2357
 TST12A  Ext                    -  2860
 UNORC   Abs   59572 #0E8B4     -  2849  2863
 VAR=1   Abs   58019 #0E2A3     -  1220  1164  1217
=VARNB-  Abs   57997 #0E28D     -  1214  1485  1488
=VARNBR  Abs   57993 #0E289     -  1213  1160  1583  1587
 XMTEST  Abs   57787 #0E1BB     -   767   742  1807
 XYEX    Ext                    -  2864
 ZATBLU  Abs   58860 #0E5EC     -  2129  1348  1496  1502  1706
```

```
 ZEROS1  Abs   57151 #0DF3F -   207   216
 ZEROS2  Abs   57162 #0DF4A -   211   214
 ZTBLU   Abs   58855 #0E5E7 -  2127
 ZTBLU+  Abs   58849 #0E5E1 -  2125  1702
 addsub  Abs   57738 #0E18A -   693   180   263
 aornan  Abs   58045 #0E2BD -  1230  1219  1224  1227  2424
 argclf  Abs   59548 #0E89C -  2793  2683  2725  2767
 ave=d1  Ext                  -   367  1622
 bSTAT   Ext                  -   292   378
 cnflc2  Abs   58195 #0E353 -  1473  1478
 cnflc3  Abs   59065 #0E6B9 -  2286  2262
 cnflct  Abs   59654 #0E906 -  2985   459  1473  2286  2857
 dv15s   Abs   58626 #0E502 -  1806   250  1513
 eIVSAR  Ext                  -  1981
 eIVSOP  Ext                  -   468  1913
 eIVSTA  Ext                  -  1874
 eSIGOP  Ext                  -  2208
 eUNORC  Ext                  -  2850
 expexc  Abs   57927 #0E247 -  1024  1638
 finita  Abs   59686 #0E926 -  3060  1214  1867  3036
 hdflt   Ext                  -  2259
 invnan  Abs   58669 #0E52D -  1909  1914  2209  2851
 ivarty  Abs   57605 #0E105 -   459   445   454
 mferr   Abs   57640 #0E128 -   470   177   381  1982
 mp15s   Abs   57779 #0E1B3 -   764   159   190   845  1419  1507  1715
 nxtst1  Abs   56932 #0DE64 -    81    76
 nxtstm  Abs   57415 #0E047 -   294    81  1658  2284
 outres  Abs   58329 #0E3D9 -  1516  2769
 outrs2  Abs   59544 #0E898 -  2769  2605  2644  2685  2687  2689  2727  2729
                              2731  2852  2861  2865
 pop2n+  Abs   59639 #0E8F7 -  2945  1477  2846  2855
 pshstk  Abs   57947 #0E25B -  1099  1161
 rclw1   Abs   57943 #0E257 -  1063   179   181
 rclw1+  Abs   57937 #0E251 -  1061   158   245
 rtncc2  Abs   57794 #0E1C2 -   770   768
 sSTAT   Ext                  -    29   139   182   283   692   697  1281  1284
                              1292
 spltA+  Abs   57659 #0E13B -   475   164   277   848
 stscr   Abs   57933 #0E24D -  1026   142   192
 subone  Abs   57735 #0E187 -   692  1356
 uMODES  Ext                  -   939
 uRES12  Ext                  -   814
=uRESD1  Abs   57838 #0E1EE -   925   264   517   819
 umode+  Abs   57884 #0E21C -   937  1158  1481  1603  1701  2909  3034
 umodeH  Abs   57880 #0E218 -   935   143
 ures12  Abs   57796 #0E1C4 -   812   931  1516  1574  1601  1606
```

Input Parameters

  Source file name is PM&STA::MS

  Listing file name is PM/STA:TI:ML::-1

  Object file name is PM%STA:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                *    M   M  N   N  &     BBBB   PPPP
                *    MM MM  N   N  & &   B   B  P   P
                *    M M M  NN  N  & &   B   B  P   P
                *    M M M  N N N   &    BBBB   PPPP
                *    M   M  N  NN  & & &  B   B  P
                *    M   M  N   N  &  &   B   B  P
                *    M   M  N   N   && &  BBBB   P
                *
                        TITLE  BEEP Module <831212.1529>
0E9F1                   ABS    #0E9F1

                ********************************************************************
                ********************************************************************
                **
                ** Name:    CLKSPD  -  Compute CPU Clockspeed
                **
                ** Category:   GENUTL
                **
                ** Purpose:
                **      Determine CPU clock speed for use in time-dependent
                **      applications.
                **
                ** Entry:
                **      None
                **
                ** Exit:
                **      Clockspeed/16 in HEX stored in CSPEED.
                **      P=0.
                **      Carry clear.
                **      INTON.
                **      HEX mode.
                **
                ** Calls:      None
                **
                ** Uses.......
                **              A,B,C[A],D0,P
                **
                ** Stk lvls:   0
                **
                ** Detail:
                **      Clockspeed is accurate to about 500 hz.
                **
                ** Algorithm:
                **      Look at nibble 1 of display driver timer.
                **      Loop until kerchunk (1/512 sec time boundary).
                **        (Since we determine kerchunk by looking at low nib
                **         only, kerchunk could not have occurred in middle
                **         of this read.)
                **      Execute 36-cycle counting loop for 32 kerchunks.
                **        (1/16 seconds).
                **      Screen for false alarm (e.g., looking for 10H and
                **        having timer kerchunk between reading nibs of
                **        20H, resulting in misreading as 10H rather than 1FH).
                **      Compute clockspeed/16 as:
                **           36*[loop counter] - 19.
```

```
                **
                ** History:
                **
                **     Date      Programmer            Modification
                **    --------   ----------   ---------------------------------
                **    05/19/82   NM           Added documentation
                **    02/23/83   NM           Smpl int to 1/16; write to CSPEED
                **
                ********************************************************************
                ********************************************************************
 0E9F1 04    =CLKSPD SETHEX
 0E9F3 20            P=      0
 0E9F5 D1            B=0     A               Counter
 0E9F7 114           A=R4
 0E9FA D0            A=0     A
 0E9FC 104           R4=A                    R4[A]=0
 0E9FF 1B00          D0=(5)  =TIMER2
       000
 0EA06 808F          INTOFF
 0EA0A 14A           A=DAT0  B               Read lower 2 nibs of timer
 0EA0D 14E    CLKS10 C=DAT0  B               Wait until kerchunk
 0EA10 902           ?A=C    P               Kchunk?
 0EA13 AF            GOYES   CLKS10          Not yet--7 cycles if fall through
 0EA15 14E           C=DAT0  B               In case kchunk in 1st digit--15 c
 0EA18 21            P=      1               Scan for another kerchunk--2 cycs
 0EA1A A0E           C=C-1   P               Time in 1/32 secs--4 cycles
 0EA1D A0E           C=C-1   P               Time in 1/16 secs--4 cycles
 0EA20 B35    CLKS20 B=B+1   X               Inc counter--6 cycles
 0EA23 14A           A=DAT0  B               Read timer again--15 cycles
 0EA26 966           ?A#C    B               Kerchunk again?--15 cycs if not
 0EA29 7F            GOYES   CLKS20          Fall thru 1/32 secs after ?a=c
 0EA2B D0            A=0     A               Kill 7 cycles
 0EA2D B35           B=B+1   X               Duplicate above loop--6 cycles
 0EA30 14A           A=DAT0  B               Read timer again--15 cycles
 0EA33 966           ?A#C    B               False alarm?--15 if yes
 0EA36 AE            GOYES   CLKS20
 0EA38 8080          INTON                   B <= FFF
 0EA3C 114           A=R4                    Will look at R4[A]
 0EA3F 8AC           ?A#0    A               Did we interrupt?
 0EA42 FA            GOYES   CLKSPD          Yes.  Recompute.
 0EA44 20            P=      0
 0EA46 C5            B=B+B   A
 0EA48 D4            A=B     A               2*B
 0EA4A F0            ASL     A               20*B
 0EA4C C5            B=B+B   A               4*B
 0EA4E C0            A=A+B   A               24*B
 0EA50 D2            C=0     A
 0EA52 3131          LCHEX   13
 0EA56 EA            A=A-C   A               24*B-13 (HEX)
 0EA58 1B00          D0=(5)  =CSPEED
       000
 0EA5F 140           DAT0=A  A
 0EA62 03            RTNCC
                ********************************************************************
                ********************************************************************
```

```
109                **
110                ** Name:(S) BEEP    -  BEEP Keyboard Execute
111                **
112                ** Category:   STEXEC
113                **
114                ** Purpose:
115                **      BEEP, BEEP ON and BEEP OFF commands from BASIC.
116                **
117                ** Entry:
118                **      Jumped on BEEP token.
119                **
120                ** Exit:
121                **      If normal exit, NXTSTM.
122                **      eDATTY if provided complex argument(s).
123                **
124                ** Calls:      BEEP: EXPEXC, POP1N, SFLAG?, BP.
125                **             BEEP ON: SFLAGC.
126                **             BEEP OFF: SFLAGS.
127                **
128                ** Detail:
129                **      BEEP ON
130                **      BEEP OFF
131                **      BEEP [ frequency [ , duration ] ]
132                **
133                ** Algorithm:
134                **      If PC points at ON token, clear BEEP disable flag.
135                **      If PC points at OFF token, set BEEP disable flag.
136                **      Else call EXPEXC;
137                **        If parameters not supplied, use default frequency
138                **        of 500 hz and default duration of 0.25 sec.
139                **        Call BP to perform beep.
140                **
141                ** History:
142                **
143                **     Date     Programmer            Modification
144                **    --------   ----------   ---------------------------------
145                **    05/20/82   NM           Added documentation
146                **
147                ********************************************************************
148                ********************************************************************
149 0EA64 0000          REL(5) =BEEPDC
           0
150 0EA69 0000          REL(5) =BEEPP
           0
151 0EA6E 14A  =BEEP    A=DAT0 B               Read next token
152 0EA71 3100          LC(2)  =tON
153 0EA75 962           ?A=C   B               Token is "ON"?
154 0EA78 A0            GOYES  LCFLB           Yes.
155 0EA7A B66           C=C+1  B               C=tOFF
156 0EA7D 966           ?A#C   B               Token is "OFF"?
157 0EA80 C1            GOYES  DOBEEP          No.
158 0EA82 3100  LCFLB   LC(2)  =flBEEP         LC Beep flag#
159 0EA86 4C0           GOC    BPON            Go if BEEP ON
160 0EA89 8E00          GOSUBL =SFLAGS         Set nobeep flag
           00
```

```
161 0EA8F 6000  NEXTST GOTO   =NEXTst        Done (to NXTSTM)
162 0EA93 8E00  BPON   GOSUBL =SFLAGC        Clear nobeep flag
          00
163 0EA99 55F           GONC   NEXTST        B.E.T.
164 0EA9C 7000  DOBEEP GOSUB  =EXPEXC        Expression execute for beep call
165 0EAA0 7000         GOSUB  =GETRG+        Get last argument
166 0EAA4 481          GOC    BEEP20         Go if no arguments
167 0EAA7 101          R1=A                  Save last argument
168 0EAAA 7000         GOSUB  =GETRG+        Get next-to-last argument
169 0EAAE 480          GOC    BEEP10         Go if only one argument
170 0EAB1 119          C=R1                  This was last arg--duration
171 0EAB4 522          GONC   BEEP30         BEEP.  B.E.T.
172 0EAB7 111   BEEP10 A=R1                  This was only arg--frequency
173 0EABA 4F0          GOC    BEEP25         Get default duration.  B.E.T.
174 0EABD AF2   BEEP20 C=0    W              Here if no args given
175 0EAC0 2E           P=     14
176 0EAC2 3250         LCHEX  205            Default frequency=500
          2
177 0EAC7 AFA          A=C    W              To A.  Fall through to get dur.
178 0EACA AF2   BEEP25 C=0    W
179 0EACD 2D           P=     13
180 0EACF 3552         LCHEX  999025         Default duration=0.25
          0999
181 0EAD7 7400  BEEP30 GOSUB  BP             Do beep.
182 0EADB 63BF         GOTO   NEXTST         Done.
183             ****************************************************************
184             ****************************************************************
185             **
186             ** Name:    BP      -  Machine-level Beep
187             ** Name:    BP+     -  Machine-level Beep
188             ** Name:(S) BP+C    -  Machine-level Beep
189             ** Name:(S) TONE    -  Machine-level Beep
190             **
191             ** Category:   GENUTL
192             **
193             ** Purpose:
194             **      Perform BEEP.
195             **
196             ** Entry:
197             **      BP: A = frequency in hz (floating point dec).
198             **          C = duration in secs (floating point dec).
199             **
200             **      BP+: A[A] = duration in msec (hex).
201             **           D[A] = frequency in hz (hex).
202             **           HEX mode.
203             **
204             **      BP+C: C[A] = duration in msec (hex).
205             **            D[A] = frequency in hz (hex).
206             **            HEX mode.
207             **
208             **      TONE: C[X] = inner loop countdown constant.
209             **            B[W] = outer loop countdown constant (# cycles).
210             **            HEX mode.
211             **            (Bypasses check of beep flag, computation of
212             **              constants based on freq, duration and
```

```
                **              clockspeed.)
                **
                **
                ** Exit:
                **      HEX mode.
                **
                ** Calls:       BP: RJUST, DCHXW, all BP+ calls.
                **              BP+: CSLW5, CSRW5, IDIV, MPY, SFLAG?.
                **
                ** Uses.......
                **              A,B,C,D,D0,P.
                **
                ** Stk lvls:    2
                **
                ** Detail:
                **      Maximum duration is 1048.575 seconds (FFFFF msec).
                **      Maximum frequency is determined by clockspeed.  At
                **        500 khz clockspeed, maximum frequency is 6757 hz.
                **
                ** Algorithm:
                **      Define: f = frequency
                **              t = duration in msec
                **              k1 = inner loop countdown constant
                **              k2 = outer loop countdown constant
                **      One beep cycle (one cycle of square wave) takes
                **      32*k1+74 machine cycles.  The routine beeps for k2 beep
                **      cycles.
                **        k1=(clkspd/f-74)/32
                **        if k1<0 then k1=0
                **        if k1>FFF then k1=FFF
                **        f'=clkspd/(32*k1+74)   {compute actual frequency}
                **        k2=f*t/1000    {compute cycle count}
                **        Execute tone loop, using k1 to time square waves
                **          and k2 to count tone cycles.
                **
                ** History:
                **
                **     Date      Programmer            Modification
                **   --------    ----------   ---------------------------------
                **   05/20/82    NM           Added documentation
                **
                ********************************************************************
                ********************************************************************
0EADF 04    =BP     SETHEX
0EAE1 B26           C=C+1  XS
0EAE4 A2E           C=C-1  XS
0EAE7 05            SETDEC
0EAE9 AF7           D=C    W              Hold duration
0EAEC 4A1           GOC    BP03           Go if duration is funny number
0EAEF 22            P=     2
0EAF1 304           LCHEX  4
0EAF4 9A3           ?D>C   XS             Exponent negative?
0EAF7 70            GOYES  BP02           Yes... convert duration to msec
0EAF9 923           ?D=C   XS             Exponent in range [400,499]?
0EAFC B0            GOYES  BP03           Yes.  Will oflo RJUST anyway.
```

```
268 0EAFE B37   BP02   D=D+1  X          Convert duration to msec.
269 0EB01 B37          D=D+1  X
270 0EB04 B37          D=D+1  X          Duration in msec
271 0EB07 8E00 BP03    GOSUBL =RJUST     Unfloat freq (error if nan)
          00
272 0EB0D AF6          C=A    W
273 0EB10 7000         GOSUB  =DCHXW     To hex
274 0EB14 AF0          A=0    W
275 0EB17 CC           A=A-1  A
276 0EB19 9F6          ?A>C   W          Frequency < FFFFF?
277 0EB1C 50           GOYES  BP05       Yes
278 0EB1E AFE          ACEX   W          No, use FFFFF
279 0EB21 AFF  BP05    CDEX   W
280 0EB24 AFA          A=C    W
281 0EB27 8E00         GOSUBL =RJUST     Unfloat duration
          00
282 0EB2D AF6          C=A    W
283 0EB30 7000         GOSUB  =DCHXW     To hex
284 0EB34 AF0          A=0    W
285 0EB37 CC           A=A-1  A
286 0EB39 9F6          ?A>C   W          Duration < FFFFF?
287 0EB3C 40           GOYES  BP+C       Yes.  Else use FFFFF.
288 0EB3E D6   =BP+    C=A    A
289 0EB40 8AB  =BP+C   ?D=0   A          Beep 0?
290 0EB43 00           RTNYES            Yes.  do nothing.
291 0EB45 7000         GOSUB  =CSLW5     Duration to C[9-5]
292 0EB49 AFF          CDEX   W          Duration to D[9-5]
293 0EB4C D5           B=C    A          Frequency to B[A]
294 0EB4E 20           P=     0
295 0EB50 3100         LC(2)  =f1BEEP
296 0EB54 8E00         GOSUBL =SFLAG?    Test nobeep flag
          00
297 0EB5A 400          RTNC              Return if set
298 0EB5D 1B00         DO=(5) =CSPEED
          000
299 0EB64 AF2          C=0    W          C[W]=frequency
300 0EB67 D9           C=B    A          D[A]=frequency
301 0EB69 D7           D=C    A
302 0EB6B AF0          A=0    W
303 0EB6E 142          A=DATO A
304 0EB71 BF0          ASL    W          A[W]=clkspeed
305 0EB74 7000         GOSUB  =IDIV      Clkspeed/frequency
306 0EB78 20           P=     0
307 0EB7A AF2          C=0    W
308 0EB7D 3166         LC(2)  102
309 0EB81 B7A          A=A-C  W          Clkspeed/freq - 102
310 0EB84 550          GONC   BP20       Go if > max frequency
311 0EB87 AF0          A=0    W          Minimum countdown const * 32
312 0EB8A BF4  BP20    ASR    W          /16
313 0EB8D B74          A=A+1  W          For rounding
314 0EB90 81C          ASRB              Countdown constant
315 0EB93 AF2          C=0    W
316 0EB96 A3E          C=C-1  X          C=0000000000000FFF
317 0EB99 9FE          ?A>=C  W          Result > max countdown const?
318 0EB9C 40           GOYES  BP30       Yes, use FFF
```

```
319 0EB9E D6          C=A    A          No, use result.
320 0EBA0 D7   BP30   D=C    A          Hold inner countdown constant.
321 0EBA2 29          P=     9
322 0EBA4 A9B         C=D    WP         C[15-10] already clear
323 0EBA7 7000        GOSUB  =CSRW5      Duration to C[A]
324 0EBAB AF0         A=0    W
325 0EBAE 142         A=DAT0 A          Clkspd/16
326 0EBB1 BF0         ASL    W          Clkspd
327 0EBB4 7000        GOSUB  =MPY       Duration * Clkspd
328 0EBB8 AF2         C=0    W
329 0EBBB DB          C=D    A          Inner loop countdown constant
330 0EBBD F2          CSL    A          *10H
331 0EBBF AF5         B=C    W
332 0EBC2 81E         CSRB              *8H
333 0EBC5 C1          B=B+C  A          *18H
334 0EBC7 81E         CSRB
335 0EBCA F2          CSL    A
336 0EBCC BF2         CSL    W          *400H
337 0EBCF B71         B=B-C  W          *(-3E8H)
338 0EBD2 BF1         BSL    W          *(-3E80H)
339 0EBD5 20          P=     0
340 0EBD7 3583        LC(6)  51000
          7C00
341 0EBDF B79         C=C-B  W          *16000+51000
342 0EBE2 7000        GOSUB  =IDIV      Outer loop countdown constant
343 0EBE6 AF8         B=A    W          Hold
344 0EBE9 DB          C=D    A
345 0EBEB 7000 =TONE  GOSUB  =CSLW5      Get inner loop const out of way
346 0EBEF AF7         D=C    W          Hold
347 0EBF2 20          P=     0
348 0EBF4 3100        LC(2)  =f1BPLD
349 0EBF8 8E00        GOSUBL =SFLAG?     Beep loud?
          00
350 0EBFE AFB         C=D    W
351 0EC01 7000        GOSUB  =CSRW5      Recover inner loop constant
352 0EC05 AFD         BCEX   W          To B[X]
353 0EC08 AF7         D=C    W          Outer loop constant to D
354 0EC0B 32F0        LCHEX  00F
          0
355 0EC10 28          P=     8
356 0EC12 540         GONC   TONE01     Go if quiet beep
357 0EC15 24          P=     4
358 0EC17 AC1  TONE01 B=0    S
359 0EC1A 860         ?ST=0  =Except
360 0EC1D 80          GOYES  TONE05
361 0EC1F 840         ST=0   =Except    Clear Exception flag for beep
362                                     (Only ATTN may set within beep)
363 0EC22 A4D         B=B-1  S          B[S]=F iff Exception flag set
364 0EC25 80F2 TONE05 CPEX   2
365 0EC29 870  TONE10 ?ST=1  =Except    Start of beep loop.  attn?
366 0EC2C F1          GOYES  TONE99     No--7 cycles if fall through
367 0EC2E 801         OUT=C             Beeper on--6
368 0EC31 AB4         A=B    X          6
369 0EC34 A3C  TONE20 A=A-1  X          6*(k+1)  (k is countdown const)
370 0EC37 5CF         GONC   TONE20     10*k+3
```

```
 371 0EC3A 80F2          CPEX   2              Toggle beeper--6
 372 0EC3E CF            D=D-1  A              More cycles?--7
 373 0EC40 58E           GONC   TONE10         Yes--10
 374               *
 375               * total period time (on & off) = 32*k+102 cycles
 376               *
 377               * Only checked D[A] field for cycle counter for speed.
 378               * In the bizarre range in which the cycle count can be
 379               * greater than FFFFF, the following code will continue
 380               * the loop with a brief hiccough every 100000H cycles.
 381               *
 382 0EC43 D3            D=0    A
 383 0EC45 A7F           D=D-1  W
 384 0EC48 50E           GONC   TONE10
 385               *
 386               * END HICCOUGH
 387               *
 388 0EC4B 949    TONE99 ?B=0   S
 389 0EC4E 50            GOYES  TON999
 390 0EC50 850           ST=1   =Except        Restore Exception flag.
 391 0EC53 8D00  TON999 GOVLNG =EXITBP
           000
 392               *-
 393               ********************************************************************
 394               ********************************************************************
 395               **
 396               ** Name:(S) CHIRP   -  Do An Annoying Little Beep
 397               **
 398               ** Category:   GENUTL
 399               **
 400               ** Purpose:
 401               **      Quick, high-pitched beep for errors and whatever.
 402               **
 403               ** Entry:
 404               **      HEX mode.
 405               **
 406               ** Exit:
 407               **      HEX mode.
 408               **
 409               ** Calls:      BP+C (falls through).
 410               **
 411               ** Uses.......
 412               **             A,B,C,D,P,DO.
 413               **
 414               ** Stk lvls:   2
 415               **
 416               ** History:
 417               **
 418               **    Date     Programmer             Modification
 419               **  --------   ----------   ----------------------------------
 420               **  08/02/83   NM           Added documentation
 421               **
 422               ********************************************************************
 423               ********************************************************************
 424 0EC5A D2    =CHIRP  C=0    A
```

```
425 0EC5C 20          P=     0
426 0EC5E 3287        LC(3)  1400          1400 hz
          5
427 0EC63 D7          D=C    A
428 0EC65 32B4        LC(3)  75            75 millisecs
          0
429 0EC6A 65DE        GOTO   BP+C
430 0EC6E             END
```

=BEEP	Abs	60014	#0EA6E	-	151			
BEEP10	Abs	60087	#0EAB7	-	172	169		
BEEP20	Abs	60093	#0EABD	-	174	166		
BEEP25	Abs	60106	#0EACA	-	178	173		
BEEP30	Abs	60119	#0EAD7	-	181	171		
BEEPDC	Ext			-	149			
BEEPP	Ext			-	150			
=BP	Abs	60127	#0EADF	-	256	181		
=BP+	Abs	60222	#0EB3E	-	288			
=BP+C	Abs	60224	#0EB40	-	289	287	429	
BP02	Abs	60158	#0EAFE	-	268	265		
BP03	Abs	60167	#0EB07	-	271	261	267	
BP05	Abs	60193	#0EB21	-	279	277		
BP20	Abs	60298	#0EB8A	-	312	310		
BP30	Abs	60320	#0EBA0	-	320	318		
BPON	Abs	60051	#0EA93	-	162	159		
=CHIRP	Abs	60506	#0EC5A	-	424			
CLKS10	Abs	59917	#0EA0D	-	75	77		
CLKS20	Abs	59936	#0EA20	-	82	85	90	
=CLKSPD	Abs	59889	#0E9F1	-	66	94		
CSLW5	Ext			-	291	345		
CSPEED	Ext			-	104	298		
CSRW5	Ext			-	323	351		
DCHXW	Ext			-	273	283		
DOBEEP	Abs	60060	#0EA9C	-	164	157		
EXITBP	Ext			-	391			
EXPEXC	Ext			-	164			
Except	Ext			-	359	361	365	390
GETRG+	Ext			-	165	168		
IDIV	Ext			-	305	342		
LCFLB	Abs	60034	#0EA82	-	158	154		
MPY	Ext			-	327			
NEXTST	Abs	60047	#0EA8F	-	161	163	182	
NEXTst	Ext			-	161			
RJUST	Ext			-	271	281		
SFLAG?	Ext			-	296	349		
SFLAGC	Ext			-	162			
SFLAGS	Ext			-	160			
TIMER2	Ext			-	72			
TON999	Abs	60499	#0EC53	-	391	389		
=TONE	Abs	60395	#0EBEB	-	345			
TONE01	Abs	60439	#0EC17	-	358	356		
TONE05	Abs	60453	#0EC25	-	364	360		
TONE10	Abs	60457	#0EC29	-	365	373	384	
TONE20	Abs	60468	#0EC34	-	369	370		
TONE99	Abs	60491	#0EC4B	-	388	366		
flBEEP	Ext			-	158	295		
flBPLD	Ext			-	348			
tON	Ext			-	152			

Input Parameters

  Source file name is MN&BP::MS

  Listing file name is MN/BP:TI:ML::-1

  Object file name is MNZBP:TI:MS::-1

```
                                  111111
                        0123456789012345
```
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
     1            *    M   M  N   N  &     U   U  TTTTT  L
     2            *    MM MM  N   N  & &   U   U    T    L
     3            *    M M M  NN  N  & &   U   U    T    L
     4            *    M M M  N N N  &     U   U    T    L
     5            *    M   M  N  NN  & & &  U   U    T    L
     6            *    M   M  N   N  &  &   U   U    T    L
     7            *    M   M  N   N   && &   UUU     T    LLLLL
     8            *
     9                       TITLE  Miscellaneous Utilities<831216.1456>
    10 0EC6E                 ABS    #0EC6E
```

```
                        STITLE IDIV, MPY and HEX/DEC conversions
                *****************************************************************
                *****************************************************************
                **
                ** Name:(S) IDIVA   -  A-field Integer Divide
                **
                ** Category:   MTHUTL
                **
                ** Purpose:
                **      Compute A/C, A mod C.
                **
                ** Entry:
                **      HEX or DEC mode according to arguments.
                **      Dividend in A[A], divisor in C[A].
                **
                ** Exit:
                **      Quotient in A[W].
                **      Remainder in B[W],C[W].
                **      Mode preserved
                **      P=15.
                **      Carry clear.
                **
                ** Calls:      IDIV (falls through).
                **
                ** Uses.......
                **              A,B,C,P
                **
                ** Stk lvls:   0
                **
                ** Algorithm:
                **      Zero out nibs 5-15 of A and C.
                **      IDIV.
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------   ----------   ---------------------------------
                **  06/22/82   NM           Added documentation
                **
                *****************************************************************
                *****************************************************************
0EC6E AF1       =IDIVA  B=0     W
0EC71 D8                B=A     A
0EC73 AF4               A=B     W
0EC76 D5                B=C     A
0EC78 AF9               C=B     W               Fall through to IDIV
                *****************************************************************
                *****************************************************************
                **
                ** Name:(S) IDIV    -  Full Word Integer Divide.
                **
                ** Category:   MTHUTL
                **
                ** Purpose:
                **      Perform HEX or DEC integer divide.
```

```
                ** 
                ** Entry:
                **      HEX or DEC mode according to arguments.
                **      Dividend in A.
                **      Divisor in C.
                ** 
                ** Exit:
                **      Quotient in A.
                **      Remainder in B and C.
                **      Mode preserved.
                **      P=15.
                **      Carry clear.
                ** 
                ** Calls:       None.
                ** 
                ** Uses.......
                **              A,B,C,P.
                ** 
                ** Stk lvls:    0
                ** 
                ** NOTE:
                **      No provision is made if called with denominator = 0.
                **      This code will get stuck in an infinite loop.  CAVEAT
                **      EMPTOR.
                ** 
                ** Algorithm:
                **      Align divisor with dividend, with P pointing at 1's
                **        digit of divisor.
                **      Divisor to B.  Clear A for result.
                **    1: While B>C do begin B=B-C W, A=A+1 P end
                **      CSR W, P=P-1.  If P wasn't zero, goto 1.
                ** 
                ** History:
                ** 
                **      Date      Programmer            Modification
                **    --------    ----------    ---------------------------------
                **    05/20/82    NM            Added documentation
                ** 
                ********************************************************************
                ********************************************************************
                * FALL THROUGH FROM ABOVE!!
OEC7B 20        =IDIV   P=      0               Find digit to start divide
OEC7D 94E        IDIV1  ?C#0    S               Flow to last dig?
OEC80 F0               GOYES   IDIV2
OEC82 9FA              ?A<=C   W
OEC85 A0               GOYES   IDIV2           Ready to divide?
OEC87 BF2              CSL     W               No, sl denominator
OEC8A 0C               P=P+1
OEC8C 50F              GONC    IDIV1           I hope this never carries
OEC8F AF8        IDIV2 B=A     W               Numerator to b
OEC92 AF0              A=0     W
OEC95 A0C              A=A-1   P
OEC98 B04        IDIV3 A=A+1   P               Inc digit of quotient
OEC9B B71        IDIV4 B=B-C   W               Dividend-divisor.  borrow?
OEC9E 59F              GONC    IDIV3           No
```

```
 121 0ECA1 A71          B=B+C  W              Yes, restore dividend.
 122 0ECA4 0D           P=P-1                 End of integer divide?
 123 0ECA6 403          GOC    MPY40          Yes.
 124 0ECA9 BF6          CSR    W              No, sr divisor
 125 0ECAC 5EE          GONC   IDIV4          Bet
 126            ****************************************************************
 127            ****************************************************************
 128            **
 129            ** Name:(S) HXDCW   -  Hex To Decimal Conversion
 130            ** Name:(S) HEXDEC  -  Hex To Decimal Conversion
 131            **
 132            ** Category:   CONVRT
 133            **
 134            ** Purpose:
 135            **      Convert a full-word HEX# or an A-field HEX # to a DEC#.
 136            **
 137            ** Entry:
 138            **   HEXDEC: Argument in A[A].
 139            **   HXDCW:  Argument in C[W] (HEX).
 140            **           Mode doesn't matter.
 141            **
 142            ** Exit:
 143            **      Result in A,B,C (DEC).
 144            **      DEC mode.
 145            **      Carry clear.
 146            **      P unaffected.
 147            **
 148            ** Calls:      MPY (falls through)
 149            **
 150            ** Uses.......
 151            **             A,B,C
 152            **
 153            ** Stk lvls:   0
 154            **
 155            ** Algorithm:
 156            **   HEXDEC: C=0 W, C=A A
 157            **   HXDCW:  A=0000000000000001
 158            **           SETDEC
 159            **           Fall through to MPY for mixed-mode multiply.
 160            **
 161            ** History:
 162            **
 163            **     Date     Programmer            Modification
 164            **   --------   ----------   ---------------------------------
 165            **   06/03/82   NM           Added documentation
 166            **   10/15/82   SA           Added HEXDEC entry point
 167            **
 168            ****************************************************************
 169            ****************************************************************
 170 0ECAF AF2  =HEXDEC C=0    W
 171 0ECB2 D6           C=A    A
 172 0ECB4 AF0  =HXDCW  A=0    W
 173 0ECB7 E4           A=A+1  A              A=1
 174 0ECB9 05           SETDEC
 175            ****************************************************************
```

```
176             ********************************************************************
177             **
178             ** Name:(S) MPY     -  HEX * HEX Or HEX * DEC Multiply.
179             **
180             ** Category:   MTHUTL
181             **
182             ** Purpose:
183             **      Perform HEX mode or mixed mode full word multiply.
184             **
185             ** Entry:
186             **      If HEX * HEX multiply:
187             **        Mode = HEX.
188             **        Arguments in A and C.
189             **
190             **      If HEX * DEC multiply:
191             **        Mode = DEC.
192             **        Hex argument in C.
193             **        Dec argument in A.
194             **
195             ** Exit:
196             **      If HEX * HEX multiply: HEX result in A,B,C.
197             **      If HEX * DEC multiply: DEC result in A,B,C.
198             **      Mode preserved.
199             **      Carry clear.
200             **      P unaffected.
201             **
202             ** Calls:      None.
203             **
204             ** Uses.......
205             **             A,B,C.
206             **
207             ** Stk lvls:   0
208             **
209             ** NOTE:
210             **      This routine provides a handy HEX to DEC conversion.
211             **      Performing a mixed-mode multiply with the hex argument
212             **      in C and a 0000000000000001 in A produces a DEC result
213             **      in C.
214             **
215             ** Algorithm:
216             **         Clear result (B).
217             **      1: CSRB.
218             **         If low bit was clear, goto 2.
219             **         Add A to result.
220             **      2: Double A.
221             **         If C#0 goto 1.
222             **         Copy result to A and C.
223             **
224             ** History:
225             **
226             **     Date     Programmer           Modification
227             **    --------  ----------   --------------------------------
228             **    05/20/82  NM           Added documentation
229             **    10/15/82  SA           Leaves result in A also.
230             **
```

```
231             ****************************************************************
232             ****************************************************************
233 0ECBB AF1   =MPY    B=0    W              Clear for result
234 0ECBE 822    MPY10  SB=0
235 0ECC1 81E           CSRB                  Srb multiplier
236 0ECC4 832           ?SB=0                 Shifted off digit?
237 0ECC7 50            GOYES  MPY20          No
238 0ECC9 A78           B=B+A  W              Yes, increment product by a
239 0ECCC A74    MPY20  A=A+A  W              Double argument in a
240 0ECCF 97E           ?C#0   W              Any more digits in multiplier?
241 0ECD2 CE            GOYES  MPY10          Yes
242 0ECD4 AF4    MPY30  A=B    W              No, result to A
243 0ECD7 AF9    MPY40  C=B    W
244 0ECDA 03            RTNCC
245             ****************************************************************
246             ****************************************************************
247             **
248             ** Name:(S) DCHXW   -  Full Word Decimal To Hex Conversion
249             **
250             ** Category:   CONVRT
251             **
252             ** Purpose:
253             **      Convert full-word DEC to full-word HEX number.
254             **
255             ** Entry:
256             **      Argument in C.
257             **      Mode doesn't matter.
258             **
259             ** Exit:
260             **      Result in A, B and C.
261             **      HEX mode.
262             **      Carry clear.
263             **      P=0.
264             **
265             ** Calls:      None.
266             **
267             ** Uses.......
268             **             A,B,C,P.
269             **
270             ** Stk lvls:   0
271             **
272             ** Algorithm:
273             **      Clear register for result.
274             **      For q=15 downto 0 do
275             **        begin
276             **          Multiply result by 10.
277             **          Add digit #q of argument to result
278             **        end.
279             **
280             ** History:
281             **
282             **     Date     Programmer            Modification
283             **   --------   ----------   ---------------------------------
284             **   06/03/82   NM           Added documentation
285             **
```

```
286             ****************************************************************
287             ****************************************************************
288 0ECDC 04    =DCHXW  SETHEX
289 0ECDE 20            P=      0
290 0ECE0 AF1           B=0     W               For result
291 0ECE3 AF0   DECH10  A=0     W
292 0ECE6 ACA           A=C     S
293 0ECE9 810           ASLC
294 0ECEC BF2           CSL     W
295 0ECEF A78           B=B+A   W               Add digit
296 0ECF2 0C            P=P+1
297 0ECF4 4FD           GOC     MPY30           Go if done
298 0ECF7 A75           B=B+B   W
299 0ECFA AF4           A=B     W
300 0ECFD A75           B=B+B   W
301 0ED00 A75           B=B+B   W
302 0ED03 A78           B=A+B   W
303 0ED06 6CDF          GOTO    DECH10
```

```
                        STITLE Register shifts
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) ASRW3   -  Shift A Right 3 Nibbles
                ** Name:(S) ASRW4   -  Shift A Right 4 Nibbles
                ** Name:(S) ASRW5   -  Shift A Right 5 Nibbles
                ** Name:(S) ASLW3   -  Shift A Left  3 Nibbles
                ** Name:(S) ASLW4   -  Shift A Left  4 Nibbles
                ** Name:(S) ASLW5   -  Shift A Left  5 Nibbles
                ** Name:(S) CSRW3   -  Shift C Right 3 Nibbles
                ** Name:(S) CSRW4   -  Shift C Right 4 Nibbles
                ** Name:(S) CSRW5   -  Shift C Right 5 Nibbles
                ** Name:(S) CSLW3   -  Shift C Left  3 Nibbles
                ** Name:(S) CSLW4   -  Shift C Left  4 Nibbles
                ** Name:(S) CSLW5   -  Shift C Left  5 Nibbles
                **
                ** Category:   GENUTL
                **
                ** Purpose:
                **      (SL or SR) (A or C) (3, 4 or 5) times.
                **
                ** Entry:
                **      Yes.
                **
                ** Exit:
                **      xSdWn: Register x shifted direction d n times.
                **      Carry and pointer unaffected.
                **
                ** Calls:      None.
                **
                ** Uses.......
                **              Register x (above, Exit conditions).
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------   ----------   ----------------------------------
                **  06/23/82   NM           Added documentation
                **
                ********************************************************************
                ********************************************************************
0ED0A BF4   =ASRW5  ASR     W
0ED0D BF4   =ASRW4  ASR     W
0ED10 BF4   =ASRW3  ASR     W
0ED13 BF4           ASR     W
0ED16 BF4           ASR     W
0ED19 01            RTN
0ED1B BF0   =ASLW5  ASL     W
0ED1E BF0   =ASLW4  ASL     W
0ED21 BF0   =ASLW3  ASL     W
0ED24 BF0           ASL     W
0ED27 BF0           ASL     W
```

```
    359 0ED2A 01           RTN
    360 0ED2C BF6  =CSRW5  CSR    W
    361 0ED2F BF6  =CSRW4  CSR    W
    362 0ED32 BF6  =CSRW3  CSR    W
    363 0ED35 BF6          CSR    W
    364 0ED38 BF6          CSR    W
    365 0ED3B 01           RTN
    366 0ED3D BF2  =CSLW5  CSL    W
    367 0ED40 BF2  =CSLW4  CSL    W
    368 0ED43 BF2  =CSLW3  CSL    W
    369 0ED46 BF2          CSL    W
    370 0ED49 BF2          CSL    W
    371 0ED4C 01           RTN
```

```
372                     STITLE WAIT
373                 ****************************************************************
374                 ****************************************************************
375                 **
376                 ** Name:    WAIT    -  WAIT Execute
377                 **
378                 ** Category:   STEXEC
379                 **
380                 ** Purpose:
381                 **      Execute WAIT statement.
382                 **
383                 ** Entry:
384                 **      Jumped on WAIT token.
385                 **
386                 ** Exit:
387                 **      NXTSTM.
388                 **      eIVARG if WAIT NaN.
389                 **
390                 ** Calls:      EXPEXC, POP1N, SEC2TK, SETALR, LSLEEP, ALMSRV,
391                 **             CKSREQ
392                 **
393                 ** Detail:
394                 **      WAIT <duration>
395                 **
396                 ** Algorithm:
397                 **      If interval negative, GOVLNG NXTSTM.
398                 **      Convert expression to hex 512ths of a second.
399                 **      Schedule alarm #5 using relative alarm set (SETALR).
400                 **   1: If alarm #5 timed out, GOVLNG NXTSTM.
401                 **      If ATTN flag set, GOVLNG NXTSTM.
402                 **      Sleep.
403                 **      CKSREQ.
404                 **      GOTO 1.
405                 **
406                 ** History:
407                 **
408                 **     Date     Programmer            Modification
409                 **    --------   ----------   ---------------------------------
410                 **    06/04/82   NM           Added documentation
411                 **
412                 ****************************************************************
413                 ****************************************************************
414 0ED4E 0000        REL(5) =WAITd
        0
415 0ED53 0000        REL(5) =WAITP
        0
416 0ED58 7782 =WAIT  GOSUB  expexc          Compute parameter
417 0ED5C 7611        GOSUB  GETRG+          Fetch it
418 0ED60 8E00        GOSUBL =SEC2TK         To hex ticks
        00
419 0ED66 978         ?A=0   W               Wait = 0?
420 0ED69 D5          GOYES  NEXTST          Yes.  no wait.
421 0ED6B 304         LCHEX  4               Alarm# of pause
422 0ED6E 8E00        GOSUBL =SETALR         Schedule wait alarm
        00
```

```
423 0ED74 8E00  WAIT10 GOSUBL =ALMSRV        Check alarm
          00
424 0ED7A 874          ?ST=1  4              WAIT alarm expired?
425 0ED7D 94           GOYES  NEXTST         Yes.
426 0ED7F 1F00         D1=(5) =ATNFLG        No, check ATTN flag.
          000
427 0ED86 1570         C=DAT1 P
428 0ED8A 90E          ?C#0   P              Attn flag set?
429 0ED8D 93           GOYES  NEXTST         Yes.  NXTSTM.
430 0ED8F 8F00         GOSBVL =LSLEEP        Go to sleep
          000
431 0ED96 8F00         GOSBVL =CKSREQ
          000
432 0ED9D 66DF         GOTO   WAIT10         Back to check for wait alarm
```

```
433                     STITLE Reset statement and CSTRST
434                 ****************************************************************
435                 ****************************************************************
436                 **
437                 ** Name:    RESET   -  Execute RESET Statement
438                 **
439                 ** Category:   STEXEC
440                 **
441                 ** Purpose:
442                 **      Execute RESET [CLOCK] statement.
443                 **
444                 ** Entry:
445                 **      D0 = PC.
446                 **      Jumped on RESET token.
447                 **
448                 ** Exit:
449                 **      Through NXTSTM.
450                 **
451                 ** Calls:      eolxck, CSTRST (if reset), RESETC (if reset
452                 **             clock).
453                 **
454                 ** Detail:
455                 **      RESET (resets flags and traps to coldstart values)
456                 **      RESET CLOCK (clears exact flag and sets AF=0)
457                 **
458                 ** History:
459                 **
460                 **    Date     Programmer              Modification
461                 **  --------   ----------   ----------------------------------
462                 **  09/21/82   NM           Added documentation
463                 **
464                 ****************************************************************
465                 ****************************************************************
466 OEDA1 0000         REL(5) =RESETd
        0
467 OEDA6 0000         REL(5) =RESETp
        0
468 OEDAB 15A1 =RESET  A=DAT0 2              Read token after RESET
469 OEDAF 8E00         GOSUBL =eolxck        EOL (or moral equivalent)?
        00
470 OEDB5 5A0          GONC   RESE10         No.
471 OEDB8 7010         GOSUB  CSTRST         Yes.  Do RESET.
472 OEDBC 6900         GOTO   NEXTST
473 OEDC0 8E00  RESE10 GOSUBL =RESETC        No.  Must be RESET CLOCK.
        00
474
475 OEDC6       =NEXTst
476 OEDC6 8C00  NEXTST GOLONG =NXTSTM
        00
477                 ****************************************************************
478                 ****************************************************************
479                 **
480                 ** Name:    CSTRST  -  Coldstart Set Flags And Traps
481                 **
482                 ** Category:   CONFIG
```

```
                **
                ** Purpose:
                **      Set system flags, user flags and arithmetic traps to
                **      coldstart setting.
                **
                ** Entry:
                **      P=0.
                **
                ** Exit:
                **      P=0.
                **      (through UPDANN)
                **
                ** Calls:      UPDANN (falls through).
                **
                ** Uses.......
                **              A[B],B[A],C,D0,P
                **
                ** Stk lvls:   1
                **
                **
                ** NOTE:
                **      Clears all user flags.
                **      Only clears lower 32 system flags (user-writeable).
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ----------------------------------
                **  09/21/82    NM           Wrote
                **
                ********************************************************************
                ********************************************************************
OEDCC 1B00 =CSTRST D0=(5) =TRPREG
      000
OEDD3 3411        LCHEX  11111
      111
OEDDA 144         DAT0=C A              Set default arithmetic traps.
OEDDD 18F         D0=D0- 16             Point at FLGREG.
OEDE0 AF2         C=0    W
OEDE3 1547        DAT0=C W              Reset all user flags.
OEDE7 18F         D0=D0- 16             Point at SYSFLG.
OEDEA 15C7        DAT0=C 8              Reset writeable system flags.
OEDEE 8C00        GOLONG =UPDANN        Update annunciators
      00
```

```
                        STITLE Window
                ********************************************************************
                ********************************************************************
                **
                ** Name:    WINDOW  -  Window Execute Routine
                **
                ** Category:   STEXEC
                **
                ** Purpose:
                **      Command to set display viewing window.
                **
                ** Entry:
                **      Jumped on WINDOW token.
                **
                ** Exit:
                **      Normal exit: NXTSTM.
                **      eIVARG if parms out of range.
                **
                ** Calls: EXPEXC, POP1-2, FLTDH.
                **
                ** Detail:
                **      WINDOW <p1> [ , <p2> ]
                **      where 1 <= p1 <= p2 <= 22.  p2 defaults to 22.
                **
                ** History:
                **
                **     Date     Programmer            Modification
                **   --------   ----------   ---------------------------------
                **   05/21/82   NM           Added documentation
                **
                ********************************************************************
                ********************************************************************
OEDF4 0000          REL(5) =WINDWd
      0
OEDF9 0000          REL(5) =WINDWp
      0
OEDFE 71E1 =WINDOW  GOSUB  expexc         Compute parameters
OEE02 7750          GOSUB  POP1-2         Pop parameters
OEE06 5A0           GONC   WIND10         Go if two parameters
OEE09 2D            P=     13
OEE0B 3322          LCHEX  1022           Default second parm to 22
      01
OEE11 108   WIND10  R0=C                  Hold second parm
OEE14 7FC1          GOSUB  FLTDH          First parm to hex
OEE18 480           GOC    WIND20         Go if no overflow and positive
OEE1B 20    IVARG   P=     0
OEE1D 6000          GOTO   =IVAERR
OEE21 120   WIND20  AROEX
OEE24 7FB1          GOSUB  FLTDH          Second parm to hex
OEE28 52F           GONC   IVARG          Go if overflow or negative
OEE2B 20    WIND30  P=     0
OEE2D D2            C=0    A
OEE2F 3161          LC(2)  22
OEE33 8B6           ?A>C   A              Second parm > 22?
OEE36 5E            GOYES  IVARG          Yes.  range error.
```

```
    576 0EE38 118          C=R0
    577 0EE3B 8B2          ?C>A   A            First parm > second parm?
    578 0EE3E DD           GOYES  IVARG        Yes.  range error.
    579 0EE40 96A          ?C=0   B            First parm zero?
    580 0EE43 8D           GOYES  IVARG        Yes.  range error.
    581 0EE45 EA           A=A-C  A            Compute windln
    582 0EE47 CE           C=C-1  A            Compute windst
    583 0EE49 1F00         D1=(5) =WINDST
              000
    584 0EE50 14D          DAT1=C B
    585 0EE53 1E00         D1=(4) =WINDLN
              00
    586 0EE59 6380         GOTO   DLAY60
```

```
                        STITLE Pop arguments
                *****************************************************************
                *****************************************************************
                **
                ** Name:    POP1-2  -  Pop 1 Or 2 Args From Result Stack
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Pop one or two args from result stack.
                **
                ** Entry:
                **      None.
                **
                ** Exit:
                **      First ARG in A.
                **      Carry set: second arg not supplied; C[W]=0.
                **      Carry clear: second arg in C.
                **      Normal ARGSTD error exits.
                **      Argument < 0 defaults to arg = 0.
                **      DECMODE.
                **
                ** Calls:      GETRG+.
                **
                ** Uses.......
                **              A,B,C,D[A],P,D1,S8-S11,R0
                **
                ** Stk lvls:   4
                **
                ** History:
                **
                **    Date      Programmer            Modification
                **   --------   ----------   ---------------------------------
                **   10/01/82   NM           Rewrote
                **
                *****************************************************************
                *****************************************************************
0EE5D 7510   POP1-2 GOSUB  GETRG+          Pop final arg.
0EE61 100           R0=A                   Stash.
0EE64 7E00          GOSUB  GETRG+          Pop 1st arg; cry set if only 1.
0EE68 118           C=R0                   Last arg to C
0EE6B 500           RTNNC                  Return if two args
0EE6E AFA           A=C    W               Only arg to A.
0EE71 AF2           C=0    W               Return 0 in C.
0EE74 01            RTN                    Carry set
                *****************************************************************
                *****************************************************************
                **
                ** Name:    GETRG+  -  Get Arg From MTHSTK W/excp Check
                **
                ** Category:   MTHSTK
                **
                ** Purpose:
                **      Pop an argument off of the math stack, with exception
                **      checking for type, NaN, projective infinity; screen
```

```
642                **      for positive arguments.
643                **
644                ** Entry:
645                **      D1=Top of stack
646                **
647                ** Exit:
648                **      Carry set if nothing on stack.
649                **      Else carry clear, 12-digit form in A, D1 incremented
650                **        by 16.  If A negative, returns A=0.
651                **      DECMODE.
652                **
653                ** Calls:       ARGSTA.
654                **
655                ** Uses.......
656                **              A,B,C,D[A],P,D0,D1,S8-S11.
657                **
658                ** Stk lvls:    3
659                **
660                ** History:
661                **
662                **    Date      Programmer              Modification
663                **  --------    ----------    ------------------------------------
664                **  07/30/82    NM            Wrote
665                **
666                ********************************************************************
667                ********************************************************************
668 0EE76 1B00 =GETRG+ D0=(5) =FORSTK
          000
669 0EE7D 142          A=DAT0 A              Bottom of mathstack
670 0EE80 137          CD1EX                 Top of mathstack
671 0EE83 135          D1=C
672 0EE86 8A2          ?A=C   A              Anything in mathstack?
673 0EE89 00           RTNYES                No.  Carry set.
674 0EE8B 7000         GOSUB  =ARGSTA        Pop with exception handling
675 0EE8F 17F          D1=D1+ 16             Point to next item
676 0EE92 948          ?A=0   S              Positive?
677 0EE95 50           GOYES  GETA10         Yes
678 0EE97 AF0          A=0    W              No.  Return 0
679 0EE9A 03    GETA10 RTNCC                 Carry clear.
```

```
                        STITLE Delay
                ********************************************************************
                ********************************************************************
                **
                ** Name:    DELAY   -  Delay Execute
                **
                ** Category:   STEXEC
                **
                ** Purpose:
                **      Execute delay statement.
                **
                ** Entry:
                **      Jumped on DELAY token.
                **
                ** Exit:
                **      NXTSTM if normal.
                **      eIVARG if arguments out of range.
                **
                ** Calls:      EXPEXC, MP32, POP1-2
                **
                ** Detail:
                **      DELAY <delayt> [ , <scrolt> ]
                **
                ** Algorithm:
                **      Delayt * 32 to hex -> DELAYT.
                **      Scrolt * 32 to hex -> SCROLT.
                **
                ** History:
                **
                **     Date     Programmer            Modification
                **   --------   ----------   ---------------------------------
                **   06/01/82   NM           Added documentation
                **
                ********************************************************************
                ********************************************************************
0EE9C 0000      0       REL(5) =DELAYd
0EEA1 0000      0       REL(5) =DELAYp
0EEA6 7931  =DELAY  GOSUB  expexc       Compute parameters
0EEAA 7FAF          GOSUB  POP1-2       Pop parameters
0EEAE 5D0           GONC   DLAY10       Go if two parameters
0EEB1 2C            P=     12
0EEB3 3652      1099    LCHEX  9990125      Default second parm to .125
                9
0EEBC 108   DLAY10  RO=C                    Hold second arg
0EEBF 7120          GOSUB  MP32             First arg*32 to A[B]
0EEC3 120           AROEX
0EEC6 7A10          GOSUB  MP32             Second arg*32 to A[B].
0EECA 118           C=RO                    First arg*32 to C[B].
0EECD 1F00  DLAY50  D1=(5) =DELAYT
                000
0EED4 14D           DAT1=C B                Write out first parm
0EED7 1E00          D1=(4) =SCROLT
```

```
          00
730 0EEDD 149   DLAY60 DAT1=A B              Write out second parm
731 0EEE0 65EE         GOTO   NEXTST
732               ****************************************************************
733               ****************************************************************
734               **
735               ** Name:    MP32    -  Multiply Fp# By 32 And To Hex
736               **
737               ** Category:   LOCAL
738               **
739               ** Purpose:
740               **      Perform computation for DELAY routine.
741               **
742               ** Entry:
743               **      A[W] = arg (floating point DEC).
744               **
745               ** Exit:
746               **      P=0.
747               **      HEX result in A[B].
748               **      FF if oflo.
749               **
750               ** Calls:      MP2-12, TRUNCC, FLTDH
751               **
752               ** Uses.......
753               **              A,B,C,D,P
754               **
755               ** Stk lvls:   2
756               **
757               ** History:
758               **
759               **     Date     Programmer             Modification
760               **   --------   ----------   ---------------------------------
761               **   06/01/82   NM           Added documentation
762               **
763               ****************************************************************
764               ****************************************************************
765 0EEE4 AF2   MP32   C=0    W
766 0EEE7 2D           P=     13
767 0EEE9 3323         LCHEX  1032
          01
768 0EEEF 05           SETDEC
769 0EEF1 8E00         GOSUBL =MP2-12        Multiply parm by 32
          00
770 0EEF7 8E00         GOSUBL =TRUNCC        Truncate to 12-digit form
          00
771 0EEFD AFA          A=C    W
772 0EF00 73E0  HEXFIX GOSUB  FLTDH          To hex
773 0EF04 500          RTNNC                 RTN if overflow
774               ****************************************************************
775               ****************************************************************
776               **
777               ** Name:    BYTE?   -  Does A[4-2] = 0?
778               **
779               ** Category:   LOCAL
780               **
```

```
                ** Purpose:
                **      Determine if A[A] fits into 1 byte.
                **
                ** Entry:
                **      HEX mode.
                **
                ** Exit:
                **      B[A] = 000A[B].
                **      If carry set, A[A] fits into one byte.
                **      Else A[A] = FFFFF.
                **
                ** Calls:       None.
                **
                ** Uses.......
                **              B[A],A[A].
                **
                ** Stk lvls:    0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ----------------------------------
                **  07/12/82    NM           Wrote
                **
                ********************************************************************
                ********************************************************************
                *
                * NOTE: ABOVE CODE FALLS THROUGH TO HERE
                *
OEF07 D1     BYTE?  B=0     A
OEF09 AE8           B=A     B
OEF0C 8A0           ?A=B    A
OEF0F 00            RTNYES
OEF11 D0            A=0     A
OEF13 CC            A=A-1   A
OEF15 03            RTNCC
```

```
                        STITLE AC/Battery check
                ********************************************************************
                ********************************************************************
                **
                ** Name:    ACBAT?  -  Check For Low Bat
                **
                ** Category:   GENUTL
                **
                ** Purpose:
                **      Determine presence of low battery condition.
                **
                ** Entry:
                **      ACBTSR: None.
                **      ACBAT?: None.
                **
                ** Exit:
                **      Carry clear.
                **      flBAT flag has been set or cleared to reflect battery
                **        condition.
                **      P=0.
                **
                ** Calls:      CKTMOU, SFLAGS, SFLAGC, WRTTMR (falls through).
                **
                ** Uses.......
                **  Exclusive: A[A],B[A],C,D[A],P,D0,D1
                **  Inclusive: D[A]
                **
                ** Stk lvls:   3
                **
                ** Detail:
                **      ACBTSR checks if TIMER3 is timed out.  If so, it falls
                **      through to ACBAT?.
                **
                ** Algorithm:
                **      ACBTSR: Return if timer3 not timed out
                **      ACBAT?: Read control nib from master driver.
                **              Wait about 100 machine cycles.
                **              Read control nib again.
                **              Set BAT annunciator iff battery low bit from
                **                control nib is set.
                **              Set timer3 for 1 minute.
                **
                ** History:
                **
                **     Date     Programmer            Modification
                **   --------   ----------   ----------------------------------
                **   06/03/82   NM           Added documentation
                **
                ********************************************************************
                ********************************************************************
 0EF17 04   =ACBTSR SETHEX
 0EF19 20           P=     0
 0EF1B 1B00         D0=(5) (=TIMER3)+5    Point at timer hinib
       000
 0EF22 7060         GOSUB  CKTMOU
```

```
871 0EF26 500          RTNNC                 Return if timer not timed out
872 0EF29 04   =ACBAT? SETHEX
873 0EF2B 1B00         D0=(5) =DD1CTL
          000
874 0EF32 1562         C=DAT0 XS             Read display control nib once
875 0EF36 26           P=     6
876 0EF38 0C    BAT05  P=P+1                 Delay about 100 machine cycles
877 0EF3A 5DF          GONC   BAT05
878 0EF3D 1562         C=DAT0 XS             Read display control nib again
879 0EF41 A26          C=C+C  XS             Set carry if low bat bit set
880 0EF44 3100         LC(2)  =f1BAT         Battery flag number
881 0EF48 5C0          GONC   BAT10          Low bat bit set?
882 0EF4B 8E00         GOSUBL =SFLAGS        Yes.  set bat flag.
          00
883 0EF51 6900         GOTO   BAT20
884 0EF55 8E00  BAT10  GOSUBL =SFLAGC        No.  clear bat flag.
          00
885 0EF5B 1B00  BAT20  D0=(5) =TIMER3
          000
886 0EF62 3700         LCHEX  80007800
          8700
          08
887 0EF6C 15C7         DAT0=C 8              Enable timer--1 minute
888 0EF70 8C00         GOLONG =WRTTMR        Now write it correctly.
          00
889            ********************************************************************
890            ********************************************************************
891            **
892            ** Name:    NOKEYS  -  Clear Keybuffer
893            **
894            ** Category:   KEYUTL
895            **
896            ** Purpose:
897            **      Clear keybuffer.
898            **
899            ** Entry:
900            **
901            ** Exit:
902            **      D0 = KEYPTR.
903            **      A[S] = 0.
904            **      Carry clear.
905            **
906            ** Calls:      None.
907            **
908            ** Uses.......
909            **             A[S],D0.
910            **
911            ** Stk lvls:   0
912            **
913            ** History:
914            **
915            **    Date     Programmer             Modification
916            **  --------   ----------   ----------------------------------
917            **  10/13/82   NM           Wrote.
918            **
```

```
                ****************************************************************
                ****************************************************************
0EF76 1B00 =NOKEYS DO=(5) =KEYPTR
      000
0EF7D AC0          A=0    S
0EF80 1504         DAT0=A S
0EF84 03           RTNCC
                ****************************************************************
                ****************************************************************
                **
                ** Name:    CKTMOU  -  Check Hinib Of Timer For Timeout
                **
                ** Category:   GENUTL
                **
                ** Purpose:
                **      Look at high nibble of timer and determine if
                **      timer has timed out.
                **
                ** Entry:
                **      HEX mode.
                **      DO points at high nibble of timer.
                **
                ** Exit:
                **      Carry set iff timed out.
                **
                ** Calls:      None.
                **
                ** Uses.......
                **              C[P] (whatever P is).
                **
                ** Stk lvls:   0
                **
                ** Detail:
                **      Handles potential problem of kerchunking during read.
                **      If timer kerchunks during read, some bits may be pulled
                **      erroneously to zero.
                **
                ** Algorithm:
                **      Read nibble at DO.  SL one bit.  RTNC (timed out).
                **      (Carry clear): perhaps we lost some bits from kchunk
                **        during the read.  Read again.  SL one bit.  RTN.
                **
                ** History:
                **
                **    Date      Programmer             Modification
                **  --------   ----------   ------------------------------------
                **  01/28/83   NM           Wrote.
                **
                ****************************************************************
                ****************************************************************
0EF86 1560 =CKTMOU C=DAT0 P
0EF8A A06          C=C+C  P              Read hinib of timer
0EF8D 400          RTNC                 RTNSC if timed out
0EF90 1560         C=DAT0 P             May have read low from kchunk
0EF94 A06          C=C+C  P             If so, this catches it
```

```
    973 0EF97 01          RTN
```

```
                        STITLE WIDTH and PWIDTH
                *****************************************************************
                *****************************************************************
                **
                ** Name:    WIDTH   -  WIDTH And PWIDTH Execute
                ** Name:    PWIDTX  -  WIDTH And PWIDTH Execute
                **
                ** Category:   STEXEC
                **
                ** Purpose:
                **      Execute WIDTH and PWIDTH statements.
                **
                ** Entry:
                **      D0 = PC.
                **      Jumped on WIDTH or PWIDTH token.
                **
                ** Exit:
                **      Through NXTSTM
                **
                ** Detail:
                **      WIDTH <new width>
                **      PWIDTH <new width>
                **
                **      If <new width> <= 1, <new width> = 1.
                **      If <new width> >= 255, <new width> = 0.
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ---------------------------------
                **  10/18/82   NM           Wrote.
                **
                *****************************************************************
                *****************************************************************
0EF99 00000             REL(5) =FIXDC
      0
0EF9E 00000             REL(5) =FIXP
      0
0EFA3 7C30 =WIDTH  GOSUB  expexc          Expr execute for width
0EFA7 7D10         GOSUB  DOWIDT          Fetch parameter
0EFAB 613F  WIDT10 GOTO   DLAY60          Write out parm & NXTSTM
                *
0EFAF 00000             REL(5) =FIXDC
      0
0EFB4 00000             REL(5) =FIXP
      0
0EFB9 7620 =PWIDTX GOSUB  expexc
0EFBD 7700         GOSUB  DOWIDT
0EFC1 1D00         D1=(2) =PWIDTH
0EFC5 45E          GOC    WIDT10          B.E.T.
                *
0EFC8 7AAE  DOWIDT GOSUB  GETRG+
0EFCC 1F00         D1=(5) =DWIDTH
      000
0EFD3 792F         GOSUB  HEXFIX
```

```
1024 0EFD7 570          GONC   MAX          Go if >255 (rtn A=0)
1025 0EFDA 96C          ?A#0   B            Arg>0?
1026 0EFDD 00           RTNYES               Yes
1027 0EFDF E4    MAX    A=A+1  A            No.  00->01 or FF->00.
1028 0EFE1 02           RTNSC
1029            *
1030 0EFE3 6000  expexc GOTO   =EXPEXC
1031 0EFE7 8C00  FLTDH  GOLONG =fltdh
           00
1032            *
```

```
1033                    STITLE FIX, SCI, ENG, STD execute
1034                ****************************************************************
1035                ****************************************************************
1036                **
1037                ** Name:    DSPF    -  FIX, SCI, ENG Statement Execute
1038                ** Name:    STD     -  Statement Execute
1039                **
1040                ** Category:   STEXEC
1041                **
1042                ** Purpose:
1043                **      Exexute FIX, SCI, ENG, STD statements
1044                **
1045                ** Entry:
1046                **      DO past FIX | SCI | ENG | STD token
1047                **        (token is reread during execute).
1048                **      P=0
1049                **
1050                ** Exit:
1051                **      Through NXTSTM
1052                **
1053                **      If STD: DSPFMT <-- 0
1054                **      If FIX: DSPFMT <-- 1
1055                **      If SCI: DSPFMT <-- 2
1056                **      If ENG: DSPFMT <-- 3
1057                **
1058                ** Detail:
1059                **      STD
1060                **      (FIX | SCI | ENG) <# digits>
1061                **
1062                ** History:
1063                **
1064                **    Date      Programmer              Modification
1065                **  --------   ----------   ----------------------------------
1066                **  07/04/82   JP           Modified documentation
1067                **  10/18/82   NM           Simplified & dummified.
1068                **
1069                ****************************************************************
1070                ****************************************************************
1071 0EFED 0000           REL(5) =STOPDC        1 Token Decompile
           0
1072 0EFF2 0000           REL(5) =RTNCC         1 Token Parse
           0
1073 0EFF7 30C   =STD     LCHEX  C              STD Display Format = 0
1074 0EFFA 7120   sfmt    GOSUB  SETFMT         Update RAM addr
1075 0EFFE 67CD           GOTO   NEXTST
1076                    *
1077 0F002 0000           REL(5) =FIXDC         FIX Decompile
           0
1078 0F007 0000           REL(5) =FIXP          FIX Parse
           0
1079 0F00C 73DF  =DSPF    GOSUB  expexc         Compute expression
1080 0F010 7130           GOSUB  SETDGT         Set display format
1081 0F014 8F00           GOSBVL =STMTST        Token # to C
           000
1082 0F01B 6EDF           GOTO   sfmt           C[0]=D, E or F for format
```

```
1083            ********************************************************************
1084            ********************************************************************
1085            **
1086            ** Name:(S) SETFMT  -  Set Display Format
1087            **
1088            ** Category:   DSPUTL
1089            **
1090            ** Purpose:
1091            **      Set FIX, SCI, ENG or STD display format.
1092            **
1093            ** Entry:
1094            **      C[0] = C for STD, D for FIX, E for SCI, F for ENG.
1095            **
1096            ** Exit:
1097            **      Carry clear.
1098            **
1099            ** Calls:      None.
1100            **
1101            ** Uses.......
1102            **  Inclusive: A[A],C[A],D[A]
1103            **
1104            ** Stk lvls:   0
1105            **
1106            ** Algorithm:
1107            **      Read DSPFMT nibble from system flags.
1108            **      Set lower 2 bits thereof.
1109            **      AND with argument passed in C[0].
1110            **      Write out DSPFMT nibble.
1111            **
1112            ** History:
1113            **
1114            **    Date     Programmer              Modification
1115            **  --------   ----------   ------------------------------------
1116            **  10/26/82   NM           Wrote.
1117            **
1118            ********************************************************************
1119            ********************************************************************
1120 0F01F D7   =SETFMT D=C    A              Hold arg
1121 0F021 133         AD1EX                  Hold D1
1122 0F024 1F00        D1=(5) =DSPFMT
           000
1123 0F02B 14F         C=DAT1 B              Read DSPFMT nib
1124 0F02E 0B          CSTEX
1125 0F030 850         ST=1   0
1126 0F033 851         ST=1   1
1127 0F036 0B          CSTEX                  Set lower 2 bits in DSPFMT nib
1128 0F038 0E67        C=C&D  B               AND in arg
1129 0F03C 15D0        DAT1=C 1               Write out new DSPFMT nib
1130 0F040 133         AD1EX                  Restore D1
1131 0F043 03          RTNCC
1132            ********************************************************************
1133            ********************************************************************
1134            **
1135            ** Name:    SETDGT  -  Set Display Digit Number
1136            **
```

```
1137                ** Category:   DSPUTL
1138                **
1139                ** Purpose:
1140                **      Set display format from number on stack.
1141                **
1142                ** Entry:
1143                **      Argument on math stack.
1144                **
1145                ** Exit:
1146                **      Carry set.
1147                **
1148                ** Calls:      GETRG+, HEXFIX.
1149                **
1150                ** Uses.......
1151                **              A,B,C,P,D0,D1.
1152                **
1153                ** Stk lvls:   4
1154                **
1155                ** History:
1156                **
1157                **    Date      Programmer              Modification
1158                **  --------    ----------   ------------------------------------
1159                **  10/19/82    NM           Wrote.
1160                **
1161                ********************************************************************
1162                ********************************************************************
1163 0F045 7D2E =SETDGT GOSUB  GETRG+          Get argument
1164 0F049 73BE        GOSUB  HEXFIX          Fix;  A=FFFFF if big.
1165 0F04D 1B00        D0=(5) =DSPDGT
           000
1166 0F054 D2          C=0    A
1167 0F056 20          P=     0
1168 0F058 30B         LCHEX  B
1169 0F05B 8B6         ?A>C   A               Arg > 11?
1170 0F05E 50          GOYES  SETF10          Yes, use 11.
1171 0F060 A86         C=A    P               No, use arg.
1172 0F063 15C0  SETF10 DAT0=C 1              Update DSPDGT | DSPFMT if STD
1173 0F067 02          RTNSC
```

```
1174                    STITLE CONTRAST execute
1175                ********************************************************************
1176                ********************************************************************
1177                **
1178                ** Name:    CNTRST  -  CONTRAST Execute
1179                **
1180                ** Category:   STEXEC
1181                **
1182                ** Purpose:
1183                **      Execute CONTRAST command.
1184                **
1185                ** Entry:
1186                **      D0 = PC.
1187                **      Jumped on CONTRAST token.
1188                **
1189                ** Exit:
1190                **      Through NXTSTM.
1191                **
1192                ** Detail:
1193                **      CONTRAST <numeric arg>
1194                **
1195                ** History:
1196                **
1197                **    Date     Programmer              Modification
1198                **  --------   ----------   ------------------------------------
1199                **  10/19/82   NM           Wrote
1200                **
1201                ********************************************************************
1202                ********************************************************************
1203 0F069 0000         REL(5) =FIXDC
           0
1204 0F06E 0000         REL(5) =FIXP
           0
1205 0F073 7C6F =CNTRST GOSUB  expexc
1206 0F077 7BFD         GOSUB  GETRG+
1207 0F07B 718E         GOSUB  HEXFIX
1208 0F07F D6           C=A    A
1209 0F081 F4           ASR    A
1210 0F083 8A8          ?A=0   A
1211 0F086 50           GOYES  CNTR10
1212 0F088 30F          LCHEX  F
1213 0F08B 1F00  CNTR10 D1=(5) =DCONTR
           000
1214 0F092 15D0         DAT1=C 1
1215 0F096 6F2D         GOTO   NEXTST
```

```
1216                    STITLE STRHDR
1217                ****************************************************************
1218                ****************************************************************
1219                **
1220                ** Name:(S) STRHDR  -  String Header
1221                **
1222                ** Category:   EXCUTL
1223                **
1224                ** Purpose:    Ensures there's enough memory to push string
1225                **             on the math stack, then writes out string header
1226                **
1227                ** Entry:      C(A)=#NIBS IN THE STRING
1228                **             D1 at top of math stack
1229                **             P=0
1230                **
1231                ** Exit:       R1[A] points to string header on stack
1232                **             D1 points past the header (where string will go)
1233                **             R1[15-5] = A[15-5] on entry.
1234                **             A[15-5] = C[15-5] on entry.
1235                **             C[A] preserved.
1236                **             Carry Clear.
1237                **           ERROR EXITS IF NOT ENOUGH MEMORY
1238                **
1239                ** Calls:      none
1240                **
1241                ** Stack lvls: 0
1242                **
1243                ** Uses:     A, C, D1, R1
1244                **
1245                ** History:
1246                **
1247                **    Date      Programmer   Modifications
1248                **  --------    ----------   -------------
1249                **  07/04/82    S.W.         Added documentation.  Modified
1250                **                            code to use AVMEME, instead of
1251                **                            TFORN, as place to push string.
1252                **  10/22/82    NM           Rewrote
1253                **
1254                ****************************************************************
1255                ****************************************************************
1256                *
1257                *
1258 0F09A 1CF    =STRHDR D1=D1- 16             Subtract header size
1259 0F09D 133          AD1EX
1260 0F0A0 EA           A=A-C  A              Proposed top-of-stack
1261 0F0A2 413          GOC    PK$E41
1262 0F0A5 109          R1=C                  Hold size
1263 0F0A8 1F00         D1=(5) =AVMEMS
          000
1264 0F0AF 147          C=DAT1 A
1265 0F0B2 8B2          ?C>A   A              Header fit?
1266 0F0B5 F1           GOYES  PK$E41         No
1267 0F0B7 131          D1=A                  Yes.  Point to start of header
1268 0F0BA 121          AR1EX                 And store in R1 (fetch str size)
1269 0F0BD AF2          C=0    W
```

```
1270 0F0C0 30F          LCHEX  F
1271 0F0C3 14D          DAT1=C B              Output string header tag byte
1272 0F0C6 D6           C=A    A
1273 0F0C8 171          D1=D1+ 2
1274 0F0CB 15DD         DAT1=C 14             Output string header size
1275 0F0CF 17D          D1=D1+ 14             Position to write string
1276 0F0D2 03           RTNCC
1277            *
1278 0F0D4 6000  PK$E41 GOTO   =NOMEM         GOTO MEMERR
1279            *
```

```
1280                    STITLE I/Obuffer autodelete
1281                ****************************************************************
1282                ****************************************************************
1283                **
1284                ** Name:    CL-IDS  -  Clear High Bit Of All I/O Buffer IDs
1285                **
1286                ** Category:   CONFIG
1287                **
1288                ** Purpose:
1289                **      Clear hibit of I/O buffer IDs prior to deepsleep
1290                **      wakeup polling.
1291                **
1292                ** Entry:
1293                **      None.
1294                **
1295                ** Exit:
1296                **      Carry set.
1297                **      HEX mode.
1298                **
1299                ** Calls:      PTBUFS, ADBSIZ.
1300                **
1301                ** Uses.......
1302                **              C,D1
1303                **
1304                ** Stk lvls:   1
1305                **
1306                ** Detail:
1307                **      Called during deepsleep wakeup prior to polling.
1308                **      This routine clears hibit of all buffer IDs.  Any
1309                **      buffers whose IDs have not been restored during polling
1310                **      are then deleted by AUTDEL.
1311                **
1312                ** Algorithm:
1313                **      Point at first buffer.
1314                **    1: Read buffer ID; return if 000 (end of list).
1315                **       Clear hibit if buffer ID (bit 11).
1316                **       Write out new buffer ID.
1317                **       Skip past buffer.
1318                **       Goto 1.
1319                **
1320                ** History:
1321                **
1322                **      Date      Programmer            Modification
1323                **    --------    ----------    ------------------------------
1324                **    06/01/82    NM            Added Documentation
1325                **
1326                ****************************************************************
1327                ****************************************************************
1328 0F0D8 7760 =CL-IDS GOSUB  PTBUFS          Point at iobfst
1329 0F0DC 147   CLID10 C=DAT1 A               Read buffer id
1330 0F0DF 93A          ?C=0   X               End of bufferlist?
1331 0F0E2 00           RTNYES
1332 0F0E4 0B           CSTEX
1333 0F0E6 84B          ST=0   11              Clear high bit
1334 0F0E9 0B           CSTEX
```

```
1335 0F0EB 1553         DAT1=C X              Write out id again
1336 0F0EF 172          D1=D1+ 3
1337 0F0F2 7A30         GOSUB  ADBSIZ         Add buffersize to get to next id
1338 0F0F6 55E          GONC   CLID10         B.E.T.
1339                 ****************************************************************
1340                 ****************************************************************
1341                 **
1342                 ** Name:    AUTDEL  -  Delete Non-restored I/O Buffers
1343                 **
1344                 ** Category:   CONFIG
1345                 **
1346                 ** Purpose:
1347                 **      Delete unneeded I/O buffers.
1348                 **
1349                 ** Entry:
1350                 **      None.
1351                 **
1352                 ** Exit:
1353                 **      Carry set.
1354                 **      HEX mode.
1355                 **
1356                 ** Calls:      PTBUFS, I/ODAL, ADBSIZ.
1357                 **
1358                 ** Uses.......
1359                 **             A,B,C,D,D0,D1,P
1360                 **
1361                 ** Stk lvls:   3
1362                 **
1363                 ** Detail:
1364                 **      Called during deepsleep wakeup after polling.  See
1365                 **      documentation for CL-IDS for details of use.
1366                 **
1367                 ** Algorithm:
1368                 **      Point at first buffer.
1369                 **   A: Read ID.
1370                 **      End of list (ID=0)?  Return if yes.
1371                 **      If ID>=800 then point at next buffer; goto A.
1372                 **      Delete buffer; goto A.
1373                 **
1374                 ** History:
1375                 **
1376                 **     Date     Programmer            Modification
1377                 **   --------   ----------   --------------------------------
1378                 **   06/01/82   NM           Added documentation
1379                 **
1380                 ****************************************************************
1381                 ****************************************************************
1382 0F0F9 7640 =AUTDEL GOSUB  PTBUFS         Point at iobfst
1383 0F0FD 147   AUTD05 C=DAT1 A              Read buffer ID (C[X])
1384 0F100 93A          ?C=0   X              End of list?
1385 0F103 00           RTNYES                Yes.
1386 0F105 0B           CSTEX                 No.  Look at ID.
1387 0F107 87B          ?ST=1  11             Hibit set?
1388 0F10A A1           GOYES  AUTD20         Yes.
1389 0F10C 0B           CSTEX                 No.
```

```
1390 0F10E D7          D=C    A              Hold buffer id
1391 0F110 137         CD1EX
1392 0F113 DF          CDEX   A              Hold pointer
1393 0F115 8E00        GOSUBL =I/ODAL        Deallocate buffer
           00
1394 0F11B DB          C=D    A
1395 0F11D 135         D1=C                  Restore buffer pointer
1396 0F120 6CDF        GOTO   AUTD05
1397 0F124 0B    AUTD20 CSTEX
1398 0F126 172         D1=D1+ 3
1399 0F129 7300        GOSUB  ADBSIZ         Point past this buffer
1400 0F12D 5FC         GONC   AUTD05         Look at next one.  B.E.T.
1401            ********************************************************************
1402            ********************************************************************
1403            **
1404            ** Name:    ADBSIZ  -  Add I/Obuffer Size To D1
1405            **
1406            ** Category:   LOCAL
1407            **
1408            ** Purpose:
1409            **      Get past I/O buffer.
1410            **
1411            ** Entry:
1412            **      D1 points at length field of I/Obuffer header.
1413            **      HEX mode.
1414            **
1415            ** Exit:
1416            **      D1 points at I/D field of next buffer.
1417            **      HEX mode.
1418            **      Carry clear.
1419            **
1420            ** Calls:      None.
1421            **
1422            ** Uses.......
1423            **  Exclusive: D1,C[A]
1424            **
1425            ** Stk lvls:   0
1426            **
1427            ** History:
1428            **
1429            **     Date     Programmer            Modification
1430            **   --------   ----------   ---------------------------------
1431            **   06/01/81   NM           Added documentation
1432            **
1433            ********************************************************************
1434            ********************************************************************
1435 0F130 D2   ADBSIZ C=0    A
1436 0F132 1573        C=DAT1 X              Read buffer length
1437 0F136 173         D1=D1+ 4              Point past buflen field
1438 0F139 133         AD1EX
1439 0F13C CA          A=A+C  A              Point to next buffer
1440 0F13E 133         AD1EX
1441 0F141 03          RTNCC
1442            ********************************************************************
1443            ********************************************************************
```

```
1444                **
1445                ** Name:    PTBUFS  -  Point At IOBFST
1446                **
1447                ** Category:   LOCAL
1448                **
1449                ** Purpose:
1450                **      Point at IOBFST+1.
1451                **
1452                ** Entry:
1453                **      None.
1454                **
1455                ** Exit:
1456                **      D1 = (IOBFST)+1.
1457                **      HEX mode.
1458                **
1459                ** Calls:      None.
1460                **
1461                ** Uses.......
1462                **  Exclusive: C[A],D1.
1463                **
1464                ** Stk lvls:   0
1465                **
1466                ** History:
1467                **
1468                **     Date     Programmer             Modification
1469                **   --------   ----------   ----------------------------------
1470                **   06/01/82   NM           Added documentation
1471                **
1472                ********************************************************************
1473                ********************************************************************
1474 OF143 04     PTBUFS SETHEX
1475 OF145 1F00           D1=(5) =IOBFST
           000
1476 OF14C 147            C=DAT1 A
1477 OF14F E6             C=C+1  A              Point past address counter
1478 OF151 135            D1=C
1479 OF154 01             RTN
1480                ********************************************************************
1481                ********************************************************************
1482                **
1483                ** Name:    RS-IDS  -  Restore Requested I/O Buffer IDs
1484                **
1485                ** Category:   CONFIG
1486                **
1487                ** Purpose:
1488                **      Restore (set hibit in) buffer IDs requested.
1489                **
1490                ** Entry:
1491                **      Call to RS-IDS is followed by a list of buffer ID's
1492                **      to be restored.  List is terminated with 000.
1493                **      See example in DETAIL (below).
1494                **
1495                ** Exit:
1496                **      Continues execution at address immediately following
1497                **      000 terminating buffer ID list.
```

```
1498                ** Carry is set.
1499                **
1500                ** Calls:      I/ORES
1501                **
1502                ** Uses.......
1503                **            A,C,D0,D1,P.
1504                **
1505                ** Stk lvls:   2
1506                **
1507                ** Detail:
1508                **      GOSBVL =RS-IDS
1509                **      CON(3) =bBUF1
1510                **      CON(3) =bBUF2
1511                **      CON(3) =bBUF3
1512                **      CON(3) =bBUF4
1513                **      CON(3) 0
1514                **      <code to be executed upon return>
1515                **
1516                ** Algorithm:
1517                **      Fetch pointer to list (from RSTK).
1518                **   1: Read 3 nibs; increment pointer 3 nibs.
1519                **      If read 000, goto 2.
1520                **      Restore buffer with ID just read.
1521                **      Goto 1.
1522                **   2: Push pointer onto RSTK; RTN.
1523                **
1524                ** History:
1525                **
1526                **     Date      Programmer            Modification
1527                **   --------   ----------   ---------------------------------
1528                **   06/02/82   NM           Wrote
1529                **
1530                ********************************************************************
1531                ********************************************************************
1532 0F156 07   =RS-IDS C=RSTK
1533 0F158 134         D0=C                    Point at first buffer ID
1534 0F15B 1563  RSID10 C=DATO X               Get buffer ID
1535 0F15F 162         D0=D0+ 3                Point past buffer ID
1536 0F162 93A         ?C=0   X                End of list?
1537 0F165 C0          GOYES  RSID20           Yes
1538 0F167 8E00        GOSUBL =I/ORES          No.  Restore this buffer.
           00
1539 0F16D 6DEF        GOTO   RSID10           On to next buffer ID.
1540 0F171 136   RSID20 CDOEX                  Fetch pointer past list
1541 0F174 06          RSTK=C
1542 0F176 01          RTN                     Jump to code after list
1543 0F178             END
```

```
=ACBAT?  Abs   61225 #0EF29 -   872
=ACBTSR  Abs   61207 #0EF17 -   867
 ADBSIZ  Abs   61744 #0F130 -  1435  1337  1399
 ALMSRV  Ext                -   423
 ARGSTA  Ext                -   674
=ASLW3   Abs   60705 #0ED21 -   356
=ASLW4   Abs   60702 #0ED1E -   355
=ASLW5   Abs   60699 #0ED1B -   354
=ASRW3   Abs   60688 #0ED10 -   350
=ASRW4   Abs   60685 #0ED0D -   349
=ASRW5   Abs   60682 #0ED0A -   348
 ATNFLG  Ext                -   426
 AUTD05  Abs   61693 #0F0FD -  1383  1396  1400
 AUTD20  Abs   61732 #0F124 -  1397  1388
=AUTDEL  Abs   61689 #0F0F9 -  1382
 AVMEMS  Ext                -  1263
 BAT05   Abs   61240 #0EF38 -   876   877
 BAT10   Abs   61269 #0EF55 -   884   881
 BAT20   Abs   61275 #0EF5B -   885   883
 BYTE?   Abs   61191 #0EF07 -   810
 CKSREQ  Ext                -   431
=CKTMOU  Abs   61318 #0EF86 -   968   870
=CL-IDS  Abs   61656 #0F0D8 -  1328
 CLID10  Abs   61660 #0F0DC -  1329  1338
 CNTR10  Abs   61579 #0F08B -  1213  1211
=CNTRST  Abs   61555 #0F073 -  1205
=CSLW3   Abs   60739 #0ED43 -   368
=CSLW4   Abs   60736 #0ED40 -   367
=CSLW5   Abs   60733 #0ED3D -   366
=CSRW3   Abs   60722 #0ED32 -   362
=CSRW4   Abs   60719 #0ED2F -   361
=CSRW5   Abs   60716 #0ED2C -   360
=CSTRST  Abs   60876 #0EDCC -   515   471
=DCHXW   Abs   60636 #0ECDC -   288
 DCONTR  Ext                -  1213
 DD1CTL  Ext                -   873
 DECH10  Abs   60643 #0ECE3 -   291   303
=DELAY   Abs   61094 #0EEA6 -   717
 DELAYT  Ext                -   727
 DELAYd  Ext                -   715
 DELAYp  Ext                -   716
 DLAY10  Abs   61116 #0EEBC -   722   719
 DLAY50  Abs   61133 #0EECD -   727
 DLAY60  Abs   61149 #0EEDD -   730   586  1012
 DOWIDT  Abs   61384 #0EFC8 -  1021  1011  1017
 DSPDGT  Ext                -  1165
=DSPF    Abs   61452 #0F00C -  1079
 DSPFMT  Ext                -  1122
 DWIDTH  Ext                -  1022
 EXPEXC  Ext                -  1030
 FIXDC   Ext                -  1008  1014  1077  1203
 FIXP    Ext                -  1009  1015  1078  1204
 FLTDH   Abs   61415 #0EFE7 -  1031   564   569   772
 FORSTK  Ext                -   668
 GETA10  Abs   61082 #0EE9A -   679   677
```

```
=GETRG+  Abs   61046 #0EE76 -   668   417   624   626  1021  1163  1206
=HEXDEC  Abs   60591 #0ECAF -   170
 HEXFIX  Abs   61184 #0EF00 -   772  1023  1164  1207
=HXDCW   Abs   60596 #0ECB4 -   172
 I/ODAL  Ext                 - 1393
 I/ORES  Ext                 - 1538
=IDIV    Abs   60539 #0EC7B -   107
 IDIV1   Abs   60541 #0EC7D -   108   114
 IDIV2   Abs   60559 #0EC8F -   115   109   111
 IDIV3   Abs   60568 #0EC98 -   118   120
 IDIV4   Abs   60571 #0EC9B -   119   125
=IDIVA   Abs   60526 #0EC6E -    52
 IOBFST  Ext                 - 1475
 IVAERR  Ext                 -  567
 IVARG   Abs   60955 #0EE1B -   566   570   575   578   580
 KEYPTR  Ext                 -  921
 LSLEEP  Ext                 -  430
 MAX     Abs   61407 #0EFDF - 1027  1024
 MP2-12  Ext                 -  769
 MP32    Abs   61156 #0EEE4 -   765   723   725
=MPY     Abs   60603 #0ECBB -   233
 MPY10   Abs   60606 #0ECBE -   234   241
 MPY20   Abs   60620 #0ECCC -   239   237
 MPY30   Abs   60628 #0ECD4 -   242   297
 MPY40   Abs   60631 #0ECD7 -   243   123
 NEXTST  Abs   60870 #0EDC6 -   476   420   425   429   472   731  1075  1215
=NEXTst  Abs   60870 #0EDC6 -   475
=NOKEYS  Abs   61302 #0EF76 -   921
 NOMEM   Ext                 - 1278
 NXTSTM  Ext                 -  476
 PK$E41  Abs   61652 #0F0D4 - 1278  1261  1266
 POP1-2  Abs   61021 #0EE5D -   624   559   718
 PTBUFS  Abs   61763 #0F143 - 1474  1328  1382
 PWIDTH  Ext                 - 1018
=PWIDTX  Abs   61369 #0EFB9 - 1016
 RESE10  Abs   60864 #0EDC0 -   473   470
=RESET   Abs   60843 #0EDAB -   468
 RESETC  Ext                 -  473
 RESETd  Ext                 -  466
 RESETp  Ext                 -  467
=RS-IDS  Abs   61782 #0F156 - 1532
 RSID10  Abs   61787 #0F15B - 1534  1539
 RSID20  Abs   61809 #0F171 - 1540  1537
 RTNCC   Ext                 - 1072
 SCROLT  Ext                 -  729
 SEC2TK  Ext                 -  418
 SETALR  Ext                 -  422
=SETDGT  Abs   61509 #0F045 - 1163  1080
 SETF10  Abs   61539 #0F063 - 1172  1170
=SETFMT  Abs   61471 #0F01F - 1120  1074
 SFLAGC  Ext                 -  884
 SFLAGS  Ext                 -  882
=STD     Abs   61431 #0EFF7 - 1073
 STMTST  Ext                 - 1081
 STOPDC  Ext                 - 1071
```

```
=STRHDR  Abs   61594 #0F09A -  1258
 TIMER3  Ext                -   869   885
 TRPREG  Ext                -   515
 TRUNCC  Ext                -   770
 UPDANN  Ext                -   523
=WAIT    Abs   60760 #0ED58 -   416
 WAIT10  Abs   60788 #0ED74 -   423   432
 WAITP   Ext                -   415
 WAITd   Ext                -   414
 WIDT10  Abs   61355 #0EFAB -  1012  1019
=WIDTH   Abs   61347 #0EFA3 -  1010
 WIND10  Abs   60945 #0EE11 -   563   560
 WIND20  Abs   60961 #0EE21 -   568   565
 WIND30  Abs   60971 #0EE2B -   571
 WINDLN  Ext                -   585
=WINDOW  Abs   60926 #0EDFE -   558
 WINDST  Ext                -   583
 WINDWd  Ext                -   556
 WINDWp  Ext                -   557
 WRTTMR  Ext                -   888
 eolxck  Ext                -   469
 expexc  Abs   61411 #0EFE3 -  1030   416   558   717  1010  1016  1079  1205
 flBAT   Ext                -   880
 fltdh   Ext                -  1031
 sfmt    Abs   61434 #0EFFA -  1074  1082
```

Input Parameters

  Source file name is MN&UTL::MS

  Listing file name is MN/UTL:TI:ML::-1

  Object file name is MN%UTL:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                    TITLE  Expression Execution Controller<831212.1511>
OF178               ABS    #OF178
               *      A    BBBB    &     EEEEE  X   X  PPPP
               *     A A   B   B  & &    E      X   X  P   P
               *    A   A  B   B  & &    E       X X   P   P
               *    A   A  BBBB    &     EEEE     X    PPPP
               *    AAAAA  B   B  & & &  E       X X   P
               *    A   A  B   B  &  &   E      X   X  P
               *    A   A  BBBB    && &  EEEEE  X   X  P
               ************************************************************
               ************************************************************
               **
               ** Name:(S) EXPEXC  -  Evaluate Expression
               ** Name:    EXPEX1  -  Evaluate Expression
               ** Name:(S) EXPEX-  -  Evaluate Expression
               ** Name:(S) EXPEX+  -  Evaluate Expression
               **
               ** Category:   EXCUTL
               **
               ** Purpose:
               **      Initiate evaluation of an expression.
               **
               ** Entry:
               **      HEX mode.
               **      D0 pointing to start of expression.
               **
               ** Exit:
               **      Carry clear.
               **      D1 pointing at top of mathstack, which contains
               **        whatever results the expressions put there.
               **      D0 pointing past expression.
               **      A[W] = 16 nibbles at top of stack (==result if this
               **        is a REAL numeric expression).
               **      If the last item in the expression was a variable,
               **        information is left in certain registers for use
               **        by the DEST routine.  See the documentation for
               **        DYNAMC and STATIC in this module.
               **
               ** Calls:      COLLAP, GETST.  Exits through EXPR.
               **
               ** Uses.......
               **             Everything available to functions:
               **             ALL CPU REGS, Function Scratch, SCRTCH,
               **
               ** Stk lvls:   4 (4 levels available to functions invoked)
               **
               ** Note:
               **      EXPEXC and EXPEX1 are different names for same entry
               **      point.
               **
               ** Algorithm:
               **   EXPEX-: Collapse mathstack to forstk.
               **           Goto expexc.
               **   EXPEX+: Save CPU status bits in STSAVE.
               **   EXPEX1:
```

```
 56            **   EXPEXC: D1 = (MTHSTK).
 57            **           Go to EXPR {i.e., evaluate expression}.
 58            **
 59            ** History:
 60            **
 61            **    Date     Programmer            Modification
 62            **  --------   ----------   ----------------------------------
 63            **             SA           Wrote
 64            **  10/13/83   NM           Attempted to document
 65            **
 66            ********************************************************************
 67            ********************************************************************
 68 0F178 8E00 =EXPEX- GOSUBL =COLLAP       Collapse Math Stack to FORSTK
          00
 69 0F17E 6110         GOTO   EXPEX2
 70 0F182 7034 =EXPEX+ GOSUB  GETST
 71 0F186      =EXPEX1
 72 0F186 1F00 =EXPEXC D1=(5) =MTHSTK       Arithmetic Stack = MTHSTK/AVMEME
          000
 73 0F18D 147          C=DAT1 A
 74 0F190 135   EXPEX2 D1=C
 75 0F193 68A0         GOTO   EXPR          GOTO EXPR
 76
 77            ********************************************************************
 78            ********************************************************************
 79            **
 80            ** Name:    BLDNUM  -  Read Tokenized Number In CPU
 81            **
 82            ** Category:   EXCUTL
 83            **
 84            ** Purpose:
 85            **      Read tokenized number at execution time.
 86            **
 87            ** Entry:
 88            **      Expression execution controller jumped on INTEGER or
 89            **        REAL constant token.
 90            **      P=0.
 91            **      Token in B[B].
 92            **      D0=PC.
 93            **      D1 = top of mathstack.
 94            **
 95            ** Exit:
 96            **      Through FNRTN1 (puts result on stack and continues
 97            **        expression execution).
 98            **
 99            ** Calls:      None.
100            **
101            ** Uses.......
102            **             Everything available to EXPEXC (since this is
103            **               part of expression execution).
104            **
105            ** Stk lvls:   4
106            **
107            ** History:
108            **
```

```
                **    Date     Programmer          Modification
                ** --------  ----------  ---------------------------------
                **            SA          Wrote
                **  10/13/83  NM          Attempted to document
                **
                ********************************************************************
                ********************************************************************
OF197 A89  =BLDNUM C=B    P             SET TEXT PTR FOR MANTISSA READ
OF19A 80F0         CPEX   0
OF19E 136          CDOEX
OF1A1 FA           C=-C   A
OF1A3 809          C+P+1
OF1A6 FA           C=-C   A
OF1A8 136          CDOEX
OF1AB 1567         C=DATO W             READ MANTISSA AND JUNK
OF1AF AC2          C=0    S             CLEAR SIGN
OF1B2 A92          C=0    WP            ZERO OUT JUNK
OF1B5 16E          DO=DO+ 15            SET TEXT POINTER PAST MANTISSA
OF1B8 21           P=     1
OF1BA 90D          ?B#0   P             FLOATER?
OF1BD 11           GOYES  FLOAT         IF SO, GOTO FLOAT
OF1BF 20           P=     0             MANUFACTURE INTEGER EXPONENT
OF1C1 3131         LCHEX  13
OF1C5 05           SETDEC
OF1C7 B69          C=C-B  B
OF1CA 6B40         GOTO   FNRTN1        GOTO FNRTN1
OF1CE 1563  FLOAT  C=DATO X             READ FLOATING EXPONENT
OF1D2 162          DO=DO+ 3             SET TEXT POINTER PAST EXPONENT
OF1D5 6040         GOTO   FNRTN1        GOTO FNRTN1

OF1D9 6000  STKERR GOTO   =NOMEM        STACK COLLISION

                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) TRMNTR  -  Process Terminator In Expr Execute
                **
                ** Category:   FNEXEC
                **
                ** Purpose:
                **      Process terminator in expression execute.  Collapse
                **      expression execution environment and return to
                **      whomever called EXPEXC.
                **
                ** Entry:
                **      D1 = mathstack pointer.
                **
                ** Exit:
                **      D1 = mathstack pointer.
                **      A[W] = 16 nibbles at top of stack.
                **
                ** Calls:      None.
                **
                ** Uses.......
                **            A,C[A].
```

```
164                **
165                ** Stk lvls:   0
166                **
167                ** History:
168                **
169                **    Date     Programmer            Modification
170                **  --------   ----------   ----------------------------------
171                **             SA           Wrote
172                **  11/01/83   NM           Attempted to document
173                **
174                ********************************************************************
175                ********************************************************************
176 0F1DD 181  =TRMNTR D0=D0- 2              BACK UP PROGRAM COUNTER
177 0F1E0 137   EXPEND CD1EX                 EXPRESSION TERMINATOR
178 0F1E3 1F00         D1=(5) =SYSEN         COLLAPSE SYSTEM STUFF
          000
179 0F1EA 143          A=DAT1 A
180 0F1ED 174          D1=D1+ 5
181 0F1F0 141          DAT1=A A
182 0F1F3 174          D1=D1+ 5
183 0F1F6 141          DAT1=A A
184 0F1F9 174          D1=D1+ 5
185 0F1FC 145          DAT1=C A              STORE MTHSTK POINTER
186 0F1FF 137          CD1EX
187 0F202 1537         A=DAT1 W              REREAD TOP OF STACK
188 0F206 03           RTNCC                 RETURN CARRY CLEAR
189
190                ********************************************************************
191                ********************************************************************
192                **
193                ** Name:    ONEDGT  -  Read Tokenized One-digit Constant
194                **
195                ** Category:   EXCUTL
196                **
197                ** Purpose:
198                **      Read tokenized one-digit constant in the course of
199                **      evaluating an expression.
200                **
201                ** Entry:
202                **      P=0.
203                **      Token# in B.
204                **      D1 at top of mathstack.
205                **      D0=PC.
206                **      Expression execution controller jumped on one-digit
207                **        constant token.
208                **
209                ** Exit:
210                **      Through FNRTN1 (falls through).
211                **
212                ** Calls:      None.
213                **
214                ** Uses.......
215                **             Everything available for expressions.
216                **
217                ** Stk lvls:   4
```

```
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------  ----------------------------------
                **             SA          Wrote
                **  10/13/83   NM          Attempted to document
                **
                ********************************************************************
                ********************************************************************
0F208 AF2       =ONEDGT C=0    W              CREATE SINGLE-DIGIT CONSTANT
0F20B A89               C=B    P
0F20E 80F0              CPEX   0
0F212 80FE              CPEX   14
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) FNRTN1  -  Function Return
                ** Name:(S) FNRTN2  -  Function Return
                ** Name:(S) FNRTN3  -  Function Return
                ** Name:(S) FNRTN4  -  Function Return
                ** Name:(S) EXPR    -  Function Return
                **
                ** Category:   EXCUTL
                **
                ** Purpose:
                **      Return to expression execution controller after
                **        evaluation of a function or operator.
                **
                ** Entry:
                **   FNRTN1: D0 = PC.
                **           D1 = stack pointer.
                **           Number to be pushed on stack in C[W].
                **   FNRTN2: A[A] = PC.
                **           D1 = stack pointer.
                **           Number to be pushed on stack in C[W].
                **   FNRTN3: A[A] = PC.
                **           D1 = new stack pointer (pointer already
                **             decremented for storing result and stack
                **             collision check already performed).
                **           Number to be pushed on stack in C[W].
                **   FNRTN4: D0 = PC.
                **           D1 = new stack pointer (pointer already
                **             decremented for storing result and stack
                **             collision check already performed).
                **           Number to be pushed on stack in C[W].
                **   EXPR:   D0 = PC.
                **           D1 = stack pointer.
                **           Result has already been put on stack.
                **
                ** Exit:
                **      Continues evaluation of expression.  Returns to
                **        whomever called expression execution controller when
                **        expression is done.
                **      Return conditions at that time:
```

```
273             **       Carry clear.
274             **       D1 at top of stack.
275             **       D0 = PC, is past expression.
276             **
277             ** Calls:      None.
278             **
279             ** Uses.......
280             **             Everything available for functions.
281             **
282             ** Stk lvls:   4
283             **
284             ** History:
285             **
286             **    Date     Programmer             Modification
287             **  --------   ----------   ----------------------------------
288             **             SA           Wrote
289             **  10/13/83   NM           Attempted to document
290             **
291             ************************************************************
292             ************************************************************
293             *
294             * Note: Above code falls into this
295             *
296 0F216 132  =FNRTN1 ADOEX                  SWAP OUT PC
297 0F219 1B00 =FNRTN2 D0=(5) =AVMEMS
        000
298 0F220 1CF          D1=D1- 16
299 0F223 137          CD1EX
300 0F226 D7           D=C    A
301 0F228 146          C=DAT0 A
302 0F22B DF           CDEX   A
303 0F22D 8BB          ?C<=D  A              STACK COLLISION?
304 0F230 9A           GOYES  STKERR         IF SO, GOTO STKERR
305 0F232 137          CD1EX
306 0F235 130  =FNRTN3 D0=A                  RESTORE PC
307 0F238 1557 =FNRTN4 DAT1=C W              PUSH NUMBER ON STACK
308 0F23C      =LPRP
309 0F23C 04   =EXPR   SETHEX                EXPRESSION INTERPRETER
310 0F23E 20           P=     0
311 0F240 D0           A=0    A
312 0F242 14A          A=DAT0 B              READ TOKEN
313 0F245 3100         LC(2)  =LASTFN
314 0F249 9E6          ?A>C   B              END OF EXPRESSION?
315 0F24C 49           GOYES  EXPEND         IF SO, GOTO EXPEND
316 0F24E D8           B=A    A              SAVE TOKEN IN B(A)
317 0F250 161          D0=D0+ 2
318 0F253 3400         LC(5)  (=MAINT)+3     MAINT ADDRESS PLUS 3
        000
319 0F25A C2           C=A+C  A              COMPUTE RELCON ADDRESS
320 0F25C A34          A=A+A  X
321 0F25F A34          A=A+A  X
322 0F262 A34          A=A+A  X
323 0F265 CA           A=A+C  A
324 0F267 132          ADOEX
325 0F26A 146          C=DAT0 A              READ RELCON
```

```
326 0F26D 132          ADOEX
327 0F270 C2           C=A+C  A              COMPUTE EXECADDRESS
328 0F272 06           RSTK=C
329 0F274 03           RTNCC                 JUMP TO FN WITH CARRY CLEAR,
330                                                P=0, HEXMODE
331
332            ****************************************************************
333            ****************************************************************
334            **
335            ** Name:    DYNAMC  -  Variable Recall
336            ** Name:    STATIC  -  Variable Recall
337            ** Name:(S) RECALL  -  Variable Recall
338            **
339            ** Category:   VARMGT
340            **
341            ** Purpose:
342            **      Recall a variable.
343            **      Also set up destination address information for
344            **      possible use by DEST after expression execution
345            **      terminates.
346            **
347            ** Entry:
348            **      P=0.
349            **      HEX mode.
350            **   STATIC: Expression execution controller jumped on variable
351            **             token (non-alpha-digit).
352            **           D0=PC.
353            **           D1=top of stack.
354            **   DYNAMC: Expression execution controller jumped on alpha-
355            **             digit variable token.
356            **           D0=PC.
357            **           D1=top of stack.
358            **   RECALL: D0=PC.
359            **           A[A]=top of stack.
360            **           D0,B[A]=address of variable register (register
361            **             contains variable if simple, else contains
362            **             dope vector).
363            **
364            ** Exit:
365            **      Through FNRTN2.
366            **      D0=PC, pointing past expression.
367            **      D1=stack pointer.
368            **      Value recalled in on top of stack.
369            **
370            ** Calls:
371            **      If we are end of expression (this recall is last thing
372            **        done): ADRS10, ADRS40, MOved3, READIN, RECADR.
373            **      If we are not at end of expression, control reverts
374            **        to expression execution controller, which could call
375            **        anything.
376            **
377            ** Uses.......
378            **             If we are not at end of expression: everything
379            **               available to expression execution controller.
380            **             If we are at end of expression: A-D,D0,D1,P.
```

```
                **
                ** Stk lvls:   4
                **            2, if we are at end of expression.
                **
                ** NOTE:
                **      This is part of expression execution.  It does not
                **      return, it goes back to the expression execution
                **      controller.  The way to use this routine is to set
                **      up the tokenized form of the variable you want to
                **      access (whether for recall or for computing the store
                **      address), complete with a terminator, point D0 at it
                **      and perform an expression execute.  You can, with some
                **      cleverness, set things up to look as though an
                **      expression execution is in progress and call this code
                **      instead of calling EXPEXC.  This might save a little
                **      execution time.
                **
                ** Detail:
                **      In addition to recalling the variable, this routine
                **      sets up information relevant to using the variable as a
                **      destination.  This information includes the variable
                **      address, substring parameters, type, array register
                **      number, maximum string length and subscript count.
                **      If this is the last thing done before the expression
                **      terminates, that information is intact upon return from
                **      the expression execution controller, and can be passed
                **      to the DEST subroutine for storage somewhere safe.
                **
                **      WHY?  One purpose of this code is to evaluate a
                **      variable on the left side of an assignment operator (=)
                **      so it can be stored into after the expression on the
                **      right side is evaluated.  DEST serves the purpose of
                **      saving the destination information so the assignment
                **      can take place later.
                **
                **      The destination information is stored in function
                **      scratch and B[W].  DEST moves it to statement scratch.
                **
                ** History:
                **
                **    Date     Programmer            Modification
                **  --------   ----------   ----------------------------------
                **             SA           Wrote
                **  10/13/83   NM           Attempted to document
                **
                ********************************************************************
                ********************************************************************
0F276 7D62  =DYNAMC GOSUB  ADRS10          CREATE CHAIN LABEL
0F27A 7DA2  =STATIC GOSUB  ADRS40          SEARCH VARIABLE CHAIN
0F27E 445           GOC    ZERO            IF NOT FOUND, GOTO ZERO
0F281 30A   =RECALL LCHEX  A               LOAD TEST DIGIT
0F284 A87           D=C    P
0F287 AC1           B=0    S               SET REAL DATA TYPE
0F28A 1567          C=DATO W               READ REGISTER
0F28E B0F           D=C-D  P               REAL NUMBER?
```

```
436 0F291 478          GOC    FNRTN2        IF SO, GOTO FNRTN2
437 0F294 80D1         P=C    1
438 0F298 880          ?P#    0             POINTER?
439 0F29B 35           GOYES  NOTNUM        IF NOT, GOTO NOTNUM
440 0F29D A0B          C=C+D  P             COMPLEX?
441 0F2A0 4C3          GOC    COMPLX        IF SO, GOTO COMPLX
442 0F2A3 E5           B=B+1  A             POINT TO THE MANT OF THE NUMBER
443 0F2A5 E5           B=B+1  A
444 0F2A7 B45  =INTSHT B=B+1  S             SET SHORT DATA TYPE
445 0F2AA 29           P=     9
446 0F2AC 187          DO=DO- 8
447 0F2AF 1567         C=DATO W             READ 5-DIGIT MANTISSA AND SIGN
448 0F2B3 A92          C=0    WP            CLEAR OUT JUNK
449 0F2B6 20           P=     0
450 0F2B8 90F          ?D#0   P             SHORT VARIABLE?
451 0F2BB D0           GOYES  SHORT         IF SO, GOTO SHORT
452 0F2BD B45          B=B+1  S             SET INTEGER DATA TYPE
453 0F2C0 70C1         GOSUB  READIN
454 0F2C4 645F         GOTO   FNRTN2        GOTO FNRTN2
455 0F2C8 16F  SHORT   DO=DO+ 16
456 0F2CB 1563         C=DATO X             READ EXPONENT
457 0F2CF 694F         GOTO   FNRTN2        GOTO FNRTN2
458
459 0F2D3 AC1  ZERO    B=0    S
460 0F2D6 AF2          C=0    W
461 0F2D9 6F3F         GOTO   FNRTN2
462
463 0F2DD 16A  COMPLX  DO=DO+ 11
464 0F2E0 146          C=DATO A             READ OFFSET
465 0F2E3 132          ADOEX
466 0F2E6 EA           A=A-C  A             COMPUTE REGISTER ADDRESS
467 0F2E8 8C00 SNARF   GOLONG =CSETUP       POLL FOR COMPLEX ROM
          00
468
469 0F2EE 89F  NOTNUM  ?P=    15            PARAMETER DATA POINTER?
470 0F2F1 92           GOYES  PASSN         IF SO, GOTO PASSN
471 0F2F3 89E          ?P=    14            PARAMETER DOPE VECTOR POINTER?
472 0F2F6 06           GOYES  PASSA         IF SO, GOTO PASSA
473 0F2F8 B45          B=B+1  S             B(S) = 8 (NUMERIC ARRAY)
474 0F2FB B45          B=B+1  S
475
476 0F2FE 75B1 VECTOR  GOSUB  RECADR        RECTIFY VECTOR ADDRESS
477 0F302 A45          B=B+B  S
478 0F305 A45          B=B+B  S
479 0F308 80D1         P=C    1             PUT SUBSCRIPT COUNT TO B(14)
480 0F30C AFD          BCEX   W              MOVE C(1) TO B(14)
481 0F30F 80FE         CPEX   14
482 0F313 AFD          BCEX   W
483 0F316 620F         GOTO   FNRTN2        GOTO FNRTN2
484            *
485 0F31A 20   PASSN   P=     0
486 0F31C A87          D=C    P             D(0)= VAR TYPE: A,B,C,D,E
487 0F31F 161          DO=DO+ 2             POINTS TO THE ADDRESS FIELD
488 0F322 146          C=DATO A             READ THE INDIRECT ADDRESS
489 0F325 D5           B=C    A             SAVE THE ADDRESS IN B(A)
```

```
490 0F327 134          DO=C
491 0F32A 30C          LCHEX  C              NOW CHECK WHAT TYPE THIS IS
492 0F32D 903          ?C=D   P              IS THIS A REAL NUMBER ?
493 0F330 E1           GOYES  PASSR          IF SO GOTO PASSR
494 0F332 30A          LCHEX  A
495 0F335 A8F          CDEX   P              C(0)=A,B,D,E   D(0)=A
496 0F338 B0F          D=C-D  P              D(0)=0,1,3,4
497 0F33B A0B          C=C+D  P
498 0F33E 132          ADOEX
499 0F341 46A          GOC    SNARF          COMPLEX NUMBER
500 0F344 132          ADOEX
501 0F347 181          DO=DO- 2
502 0F34A 6C5F         GOTO   INTSHT         IS EITHER INTEGER OR SHORT REAL
503 0F34E 1567 PASSR   C=DATO W              JUST READ THE REAL NUMBER
504 0F352 66CE         GOTO   FNRTN2
505           * THE INDIRECT ADDRESS IS POINTING TO A DOPE VECTOR OF AN ARRAY
506 0F356 161  PASSA   DO=DO+ 2
507 0F359 146          C=DATO A              READ THE INDIRECT ADDR
508 0F35C D5           B=C    A              SAVE THE DOPE VECTOR ADDR IN B
509 0F35E 134          DO=C
510 0F361 1567         C=DATO W              READ THE DOPE VECTOR
511 0F365 80D1         P=C    1
512 0F369 AC1          B=0    S
513 0F36C 618F         GOTO   NOTNUM
514           *
515 0F370 6D8F VECTLK  GOTO   VECTOR
516           *
517 0F374 DC    NULL   ABEX   A              LEAVE PC IN B-REGISTER
518 0F376 130          DO=A                  LEAVE CHAIN LABEL IN DO
519 0F379 1CF          D1=D1- 16             CREATE NULL STRING ON STACK
520 0F37C 3400         LC(5)  =AVMEMS
          000
521 0F383 137          CD1EX
522 0F386 143          A=DAT1 A
523 0F389 8BE          ?A>=C  A              STACK COLLISION?
524 0F38C 36           GOYES  STKLNK         IF SO, GOTO STKERR
525 0F38E 135          D1=C
526 0F391 17F          D1=D1+ 16
527 0F394 D2           C=0    A              REGISTER NUMBER ZERO
528 0F396 D3           D=0    A              ACTUAL LENGTH ZERO
529 0F398 66B0         GOTO   STRHDR         GOTO STRHDR
530 0F39C 7741 =STRING GOSUB  ADRS10         SEARCH VARIABLE CHAIN
531 0F3A0 7781         GOSUB  ADRS40
532 0F3A4 4FC          GOC    NULL           IF NOT FOUND, GOTO NULL
533 0F3A7 1567 =PTRRCL C=DATO W              READ REGISTER AT ADDRESS
534 0F3AB 80D1         P=C    1
535 0F3AF 890          ?P=    0              SIMPLE STRING?
536 0F3B2 14           GOYES  $IMPLE         IF SO, GOTO $IMPLE
537 0F3B4 AC1          B=0    S              B(S) WILL EQUAL TO "C" LATER TO
538 0F3B7 B45          B=B+1  S               INDICATE THIS IS A STRING VECTOR
539 0F3BA B45          B=B+1  S
540 0F3BD B45          B=B+1  S
541 0F3C0 891          ?P=    1              ARRAY POINTER?
542 0F3C3 DA           GOYES  VECTLK         IF SO, GOTO VECTOR
543 0F3C5 89E          ?P=    14             INDIRECT PASSING DOPE VECT ADDR?
```

```
544 0F3C8 A1          GOYES  STRVCT        IF SO GOTO STRVCT
545 0F3CA 166         DO=DO+ 7
546 0F3CD D2          C=0    A
547 0F3CF 15E3        C=DATO 4              READ THE STRING MAXIMUM LENGTH
548 0F3D3 D7          D=C    A              SAVE THE MAX.LEN IN D
549 0F3D5 184         DO=DO- 5              POINTS BACK TO STRING ADDRESS
550 0F3D8 146         C=DATO A
551 0F3DB 134         DO=C
552 0F3DE 6D20        GOTO   STRRCL         UNCONDITIONAL GOTO STRRCL
553 0F3E2 161  STRVCT DO=DO+ 2
554 0F3E5 146         C=DATO A              READ THE DOPE FACTOR ADDR
555 0F3E8 134         DO=C
556 0F3EB 6BBF        GOTO   PTRRCL
557 0F3EF 69ED STKLNK GOTO   STKERR         SHORT JUMP LINKAGE
558 0F3F3 162  $IMPLE DO=DO+ 3
559 0F3F6 D2          C=0    A
560 0F3F8 15E3        C=DATO 4              READ MAXIMUM LENGTH
561 0F3FC D7          D=C    A
562 0F3FE 167         DO=DO+ 8
563 0F401 146         C=DATO A
564 0F404 132         ADOEX
565 0F407 EA          A=A-C  A              RECTIFY ADDRESS
566 0F409 132         ADOEX
567 0F40C D8  =STRRCL B=A    A              SAVE PC IN B
568 0F40E D2          C=0    A
569 0F410 15E3        C=DATO 4              READ LENGTH OF STRING TEXT
570 0F414 DF          CDEX   A
571 0F416 06          RSTK=C                SAVE MAX LENGTH ON HARDWARE STACK
572 0F418 C7          D=D+D  A
573 0F41A 3400        LC(5)  =AVMEMS
            000
574 0F421 136         CDOEX
575 0F424 CB          C=C+D  A              COMPUTE ADDRESS OF END OF STRING
576 0F426 142         A=DATO A              READ AVAILABLE MEMORY POINTER
577 0F429 134         DO=C
578 0F42C 163         DO=DO+ 4
579 0F42F 1CF         D1=D1- 16
580 0F432 137         CD1EX
581 0F435 135         D1=C
582 0F438 17F         D1=D1+ 16
583 0F43B E2          C=C-A  A              COMPUTE FREE STACK SPACE
584 0F43D 8BB         ?C<=D  A              STACK COLLISION?
585 0F440 FA          GOYES  STKLNK         IF SO, GOTO STKERR
586 0F442 DB          C=D    A
587 0F444 8E00        GOSUBL =MOved3        RECALL STRING TO STACK
            00
588 0F44A 183         DO=DO- 4
589 0F44D 07          C=RSTK                RETRIEVE MAXIMUM LENGTH
590 0F44F 1C3  STRHDR D1=D1- 4
591 0F452 15D3 =FAKE  DAT1=C 4              WRITE MAXLEN/REGNUM TO STACK
592 0F456 2A          P=     10
593 0F458 A92         C=0    WP             CONSTRUCT ADDRESS RESIDUAL
594 0F45B 35EF        LCHEX  DFFFFE
            FFFD
595 0F463 D9          C=B    A
```

```
596 0F465 136          CDOEX
597 0F468 AF5          B=C    W
598 0F46B 1C4          D1=D1- 5
599 0F46E 145          DAT1=C A              WRITE ADDRESS TO STACK
600 0F471 DB           C=D    A
601 0F473 BF2          CSL    W              CONSTRUCT STRING INDICATOR
602 0F476 812          CSLC
603 0F479 1C6          D1=D1- 7
604 0F47C 15D6         DAT1=C 7              WRITE INDICATOR & LENGTH TO STACK
605 0F480 6BBD         GOTO   EXPR           REPEAT EXPR
606             ********************************************************************
607             ********************************************************************
608             **
609             ** Name:(S) READIN  -  Read Something In
610             **
611             ** Category:   MTHUTL
612             **
613             ** Purpose:
614             **      Probably.
615             **
616             ** Entry:
617             **      Unclear.
618             **
619             ** Exit:
620             **      Unclear.
621             **
622             ** Calls:      None.
623             **
624             ** Uses.......
625             **              D,P,C[S].
626             **
627             ** Stk lvls:   0
628             **
629             ** History:
630             **
631             **    Date      Programmer             Modification
632             **  --------    ----------   ----------------------------------
633             **              SA           Wrote
634             **  11/01/83    NM           Attempted to document
635             **
636             ********************************************************************
637             ********************************************************************
638 0F484 AC7  =READIN D=C    S              UNPACK SIGN & EXPONENT
639 0F487 2F           P=     15
640 0F489 3385         LCHEX  F058
          0F
641 0F48F 20           P=     0
642 0F491 0E47         C=C&D  S
643 0F495 B43          D=D-C  S
644 0F498 AE3          D=0    B
645 0F49B 813          DSLC
646 0F49E ABF          CDEX   X
647 0F4A1 B03          D=D-C  P              NAN OR INF?
648 0F4A4 580          GONC   RDIN10         IF NOT, GOTO RDIN10
649 0F4A7 B8F          D=-D-1 P              CONSTRUCT PROPER EXPONENT
```

```
 650 0F4AA ABB          C=D    X
 651 0F4AD 94A   RDIN10 ?C=0   S              POSITIVE NUMBER?
 652 0F4B0 00           RTNYES                 IF SO, RETURN
 653 0F4B2 B46          C=C+1  S              CONSTRUCT PROPER SIGN DIGIT
 654 0F4B5 01           RTN                   RETURN
 655             *
 656             ********************************************************************
 657             ********************************************************************
 658             **
 659             ** Name:(S) RECADR  -  Some Recall Utility
 660             **
 661             ** Category:   VARMGT
 662             **
 663             ** Purpose:
 664             **      Perform D0:=D0+11 ; C[9-5]:=D0-C[9-5].  Evidently
 665             **      useful for recalling things.
 666             **
 667             ** Entry:
 668             **      Things in C and D0.
 669             **      HEX mode.
 670             **
 671             ** Exit:
 672             **      D0 has been incremented by 11.
 673             **      C[9-5] = New D0 - C[9-5].
 674             **      HEX mode.
 675             **
 676             ** Calls:      0
 677             **
 678             ** Uses.......
 679             **              D0,C[9-5].
 680             **
 681             ** Stk lvls:   0
 682             **
 683             ** History:
 684             **
 685             **    Date      Programmer              Modification
 686             **  --------    ----------   ---------------------------------
 687             **              SR           Wrote
 688             **  11/09/83    NM           Attempted to document
 689             **
 690             ********************************************************************
 691             ********************************************************************
 692 0F4B7 8E00 =RECADR GOSUBL =Cslc5
         00
 693 0F4BD 16A          D0=D0+ 11
 694 0F4C0 132          ADOEX
 695 0F4C3 EE           C=A-C  A
 696 0F4C5 132          ADOEX
 697 0F4C8 8D00 =csrc5  GOVLNG =CSRC5
         000
 698
 699             ********************************************************************
 700             ********************************************************************
 701             **
 702             ** Name:(S) ADRSUB  -  Get Variable Name From Token Stream
```

```
                **
                ** Category:   VARMGT
                **
                ** Purpose:
                **      Read a token stream for a variable and return 3-digit
                **      code for that variable
                **
                ** Entry:
                **      P=0.
                **      HEX mode.
                **      D0 points at token stream
                **
                ** Exit:
                **      P=0.
                **      B(X) = 3-digit code for variable
                **           (Defining aa = ASCII code for variable name)
                **           = 0aa if simple variable.
                **           = qaa if alpha-digit variable, where q = digit+1.
                **           = 0bb if string var, where bb = aa ! 20H.
                **           = qbb if alpha-digit string var.
                **      D0 points past last byte of variable tokenization.
                **      Carry set
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: B(X),C(X),D0.
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   -----------------------------------
                **              SA           Wrote
                **  10/13/83    NM           Attempted to document
                **
                ********************************************************************
                ********************************************************************
0F4CF D1    =ADRSUB B=0     A               INITIALIZE
0F4D1 14E           C=DATO B                READ TOKEN
0F4D4 161           D0=D0+ 2
0F4D7 AE5           B=C     B               SAVE IN B-REGISTER
0F4DA 3102          LCHEX   20
0F4DE 0E65          C=B&C   B
0F4E2 96A           ?C=0    B               ALPHADIGIT OR STRING VARIABLE?
0F4E5 00            RTNYES                  IF NOT, RETURN
0F4E7 BE1    ADRS10 BSL     B
0F4EA BB1           BSL     X
0F4ED 14E           C=DATO B
0F4F0 161           D0=D0+ 2
0F4F3 AE5           B=C     B
0F4F6 3206          LCHEX   D60
          D
0F4FB 925           ?B#C    XS              STRING VARIABLE?
```

```
757 0F4FE 42          GOYES  ADRS30        IF NOT, GOTO ADRS30
758 0F500 AA1         B=0    XS
759 0F503 A2D         B=B-1  XS            B(XS) = F
760 0F506 9E5         ?B<C   B             ALPHADIGIT STRING?
761 0F509 51          GOYES  ADRS20        IF NOT, GOTO ADRS20
762 0F50B BE1         BSL    B
763 0F50E BB1         BSL    X
764 0F511 14E         C=DATO B
765 0F514 161         DO=DO+ 2
766 0F517 AE5         B=C    B
767 0F51A 3106        LCHEX  60
768 0F51E 0E69 ADRS20 B=B!C  B
769 0F522 B25  ADRS30 B=B+1  XS
770 0F525 02          RTNSC
771           ****************************************************************
772           ****************************************************************
773           **
774           ** Name:(S) ADDRSS  -  Find Address Of A Variable
775           ** Name:(S) ADRS40  -  Find Address Of A Variable
776           ** Name:(S) ADRS50  -  Find Address Of Var Not Of Parm Chain
777           ** Name:(S) FIND    -  Find Address Of Var Not Of Parm Chain
778           ** Name:(S) ADRS80  -  Find Address Of Var Not Of Parm Chain
779           **
780           ** Category:   VARMGT
781           **
782           ** Purpose:
783           **   ADDRSS, ADRS40: Search parameter chain and then variable
784           **                   chains to find a variable.
785           **   ADRS50: Search variable chains to find variable (do not
786           **           search parameter chain.
787           **   FIND  : Same as ADRS50 except search already in progress.
788           **   ADRS80: Same as FIND except DATO already read.
789           **
790           ** Entry:
791           **      P=0.
792           **      ADDRSS:  DO points at token stream of variable to be
793           **                   found.
794           **      ADRS40:  B[X] contains 3-digit code for variable to be
795           **                   found.
796           **      ADRS50:  B[X] contains 3-digit code for variable to be
797           **                   found.
798           **      FIND  :  Search already in progress.  B[X] as above.
799           **               DO points at a variable name entry in variable
800           **                 chain.
801           **               D[B] = #entries left in chain.
802           **      ADRS80:  Same as FIND + C[X] contains entry already
803           **                 read at DO.
804           **
805           ** Exit:
806           **      P      = 0
807           **      Carry set if variable not found
808           **      Carry clear if variable found
809           **             DO,B(A) = Address of variable register
810           **      A[A] = DO at time of entry (if ADRS40 called).
811           **             Pointer past variable tokenization (if ADDRSS
```

```
812                **                              called).
813                **              A[A] at time of entry (if ADRS50, ADRS80
814                **                              called).
815                **
816                ** Calls:       CHNHED,ADRS70,ADDRSS calls ADRSUB
817                **
818                ** Uses.......
819                **              D0,A(A),B(A),C(6-0),D(A)
820                **
821                ** Stk lvls:    1
822                **
823                ** Detail:
824                **      First searches parameter chain for variable (in case
825                **      passed in CALL).  Then searches variable chain.
826                **
827                ** History:
828                **
829                **    Date      Programmer              Modification
830                **  --------    ----------   ----------------------------------
831                **              SA           Wrote
832                **  10/13/83    NM           Attempted to document
833                **
834                ********************************************************************
835                ********************************************************************
836 0F527 74AF  =ADDRSS GOSUB  ADRSUB
837 0F52B 132   =ADRS40 ADOEX
838 0F52E 1B00          D0=(5) =PRMPTR
          000
839 0F535 14E           C=DAT0 B              READ PARAMETER CHAIN POINTER
840 0F538 A6E           C=C-1  B              NULL CHAIN?
841 0F53B 451           GOC    ADRS50         IF SO, GOTO ADRS50
842 0F53E AE7           D=C    B              PREPARE FOR PARM CHAIN SEARCH
843 0F541 161           D0=D0+ 2
844 0F544 146           C=DAT0 A              C = PRMPTR
845 0F547 134           D0=C
846 0F54A 7F00          GOSUB  ADRS70         SEARCH PARAMETER CHAIN
847 0F54E 500           RTNNC                 IF FOUND, RETURN CARRY CLEAR
848 0F551 7420  =ADRS50 GOSUB  CHNHED         GET CHAIN HEAD POINTER
849 0F555 5D0           GONC   FIND           IF CHAIN NOT NULL, GOTO FIND
850 0F558 02            RTNSC                 RETURN CARRY SET
851 0F55A 16F    ADRS60 D0=D0+ 16             POINT TO NEXT CHAIN ELEMENT
852 0F55D A6F    ADRS70 D=D-1  B              CHAIN EXHAUSTED?
853 0F560 400           RTNC                  IF SO, RETURN CARRY SET
854 0F563 1563  =FIND   C=DAT0 X              READ CHAIN LABEL
855 0F567 162   =ADRS80 D0=D0+ 3              POINT TO REGISTER CONTENTS
856 0F56A 935           ?B#C   X              VARIABLE FOUND?
857 0F56D DE            GOYES  ADRS60         IF NOT, REPEAT ADRS60
858 0F56F 132           ADOEX
859 0F572 D8            B=A    A              STORE ADDRESS IN B
860 0F574 132           ADOEX
861 0F577 03            RTNCC                 RETURN CARRY CLEAR
862                ********************************************************************
863                ********************************************************************
864                **
865                ** Name:(S) CHNHED  -  Point To Variable Chain Head
```

```
866             **
867             ** Category:   VARMGT
868             **
869             ** Purpose:
870             **      Point to variable chain head and return # entries in
871             **      chain.
872             **
873             ** Entry:
874             **      P=0.
875             **      HEX mode.
876             **      B[X] = three-digit variable name (see ADRSUB doc hdr).
877             **
878             ** Exit:
879             **      P=0.
880             **      HEX mode.
881             **      D[B]=# items in chain - 1.
882             **      Carry set iff chain empty.
883             **      C[A], DO=pointer to chain head.
884             **
885             ** Calls:      None.
886             **
887             ** Uses.......
888             **              C[A],C[6-0],DO.
889             **
890             ** Stk lvls:   0
891             **
892             ** History:
893             **
894             **    Date      Programmer              Modification
895             **  --------    ----------   ------------------------------------
896             **              SA           Wrote
897             **  10/13/83    NM           Attempted to document
898             **
899             ****************************************************************
900             ****************************************************************
901 0F579 D2  =CHNHED C=0    A               RECONSTRUCT CHAIN NAME
902 0F57B 31F5         LCHEX  5F
903 0F57F 0E65         C=B&C  B
904 0F583 D7           D=C    A               COMPUTE ADDRESS OF POINTER
905 0F585 A66          C=C+C  B
906 0F588 A33          D=C+D  X
907 0F58B A36          C=C+C  X
908 0F58E A33          D=C+D  X
909 0F591 3400         LC(5)  (=CHNLST)-\A\*7
        000
910 0F598 CB           C=C+D  A
911 0F59A 134          DO=C
912 0F59D 15E6         C=DATO 7               READ POINTER & COUNTER
913 0F5A1 AE7          D=C    B
914 0F5A4 26           P=     6
915 0F5A6 B96          CSR    WP
916 0F5A9 B96          CSR    WP
917 0F5AC 134          DO=C                   LOAD CHAIN HEAD ADDRESS
918 0F5AF 20           P=     0
919 0F5B1 A6F          D=D-1  B               COND'TN CTR, NULL CHN SETS CARRY
```

```
920 0F5B4 01          RTN                     RETURN
921           ************************************************************
922 0F5B6 0B  =GETST  CSTEX
923 0F5B8 1F00        D1=(5) =STSAVE          NEED 5 NIB LOAD HERE (SW-6/28/82)
          000
924 0F5BF 1553        DAT1=C X                SAVE PREVIOUS STATUS BITS
925 0F5C3 03          RTNCC                   RETURN CARRY CLEAR
926           ************************************************************
927           ************************************************************
928           **
929           ** Name:(S) RSTST   -  Restore Status Bits
930           **
931           ** Category:   MTHUTL
932           **
933           ** Purpose:
934           **      Restore status bits saved in STSAVE.
935           **
936           ** Entry:
937           **      None.
938           **
939           ** Exit:
940           **      Status bits restored.
941           **      Carry clear.
942           **
943           ** Calls:      None.
944           **
945           ** Uses.......
946           **            A[A],C[X].
947           **
948           ** Stk lvls:   0
949           **
950           ** History:
951           **
952           **    Date     Programmer              Modification
953           **  --------   ----------   ----------------------------------
954           **             SA           Wrote
955           **  11/01/83   NM           Added documentation
956           **
957           ************************************************************
958           ************************************************************
959 0F5C5 132 =RSTST  ADOEX
960 0F5C8 1B00        D0=(5) =STSAVE
          000
961 0F5CF 1563        C=DAT0 X
962 0F5D3 0B          CSTEX
963 0F5D5 132         ADOEX
964 0F5D8 03          RTNCC
965 0F5DA             END
```

$IMPLE	Abs	62451	#0F3F3	-	558	536					
=ADDRSS	Abs	62759	#0F527	-	836						
ADRS10	Abs	62695	#0F4E7	-	750	428	530				
ADRS20	Abs	62750	#0F51E	-	768	761					
ADRS30	Abs	62754	#0F522	-	769	757					
=ADRS40	Abs	62763	#0F52B	-	837	429	531				
=ADRS50	Abs	62801	#0F551	-	848	841					
ADRS60	Abs	62810	#0F55A	-	851	857					
ADRS70	Abs	62813	#0F55D	-	852	846					
=ADRS80	Abs	62823	#0F567	-	855						
=ADRSUB	Abs	62671	#0F4CF	-	742	836					
AVMEMS	Ext			-	297	520	573				
=BLDNUM	Abs	61847	#0F197	-	116						
=CHNHED	Abs	62841	#0F579	-	901	848					
CHNLST	Ext			-	909						
COLLAP	Ext			-	68						
COMPLX	Abs	62173	#0F2DD	-	463	441					
CSETUP	Ext			-	467						
CSRC5	Ext			-	697						
Cslc5	Ext			-	692						
=DYNAMC	Abs	62070	#0F276	-	428						
EXPEND	Abs	61920	#0F1E0	-	177	315					
=EXPEX+	Abs	61826	#0F182	-	70						
=EXPEX-	Abs	61816	#0F178	-	68						
=EXPEX1	Abs	61830	#0F186	-	71						
EXPEX2	Abs	61840	#0F190	-	74	69					
=EXPEXC	Abs	61830	#0F186	-	72						
=EXPR	Abs	62012	#0F23C	-	309	75	605				
=FAKE	Abs	62546	#0F452	-	591						
=FIND	Abs	62819	#0F563	-	854	849					
FLOAT	Abs	61902	#0F1CE	-	135	129					
=FNRTN1	Abs	61974	#0F216	-	296	134	137				
=FNRTN2	Abs	61977	#0F219	-	297	436	454	457	461	483	504
=FNRTN3	Abs	62005	#0F235	-	306						
=FNRTN4	Abs	62008	#0F238	-	307						
=GETST	Abs	62902	#0F5B6	-	922	70					
=INTSHT	Abs	62119	#0F2A7	-	444	502					
LASTFN	Ext			-	313						
=LPRP	Abs	62012	#0F23C	-	308						
MAINT	Ext			-	318						
MOved3	Ext			-	587						
MTHSTK	Ext			-	72						
NOMEM	Ext			-	139						
NOTNUM	Abs	62190	#0F2EE	-	469	439	513				
NULL	Abs	62324	#0F374	-	517	532					
=ONEDGT	Abs	61960	#0F208	-	228						
PASSA	Abs	62294	#0F356	-	506	472					
PASSN	Abs	62234	#0F31A	-	485	470					
PASSR	Abs	62286	#0F34E	-	503	493					
PRMPTR	Ext			-	838						
=PTRRCL	Abs	62375	#0F3A7	-	533	556					
RDIN10	Abs	62637	#0F4AD	-	651	648					
=READIN	Abs	62596	#0F484	-	638	453					
=RECADR	Abs	62647	#0F4B7	-	692	476					
=RECALL	Abs	62081	#0F281	-	431						

```
=RSTST   Abs   62917 #0F5C5 -   959
 SHORT   Abs   62152 #0F2C8 -   455   451
 SNARF   Abs   62184 #0F2E8 -   467   499
=STATIC  Abs   62074 #0F27A -   429
 STKERR  Abs   61913 #0F1D9 -   139   304   557
 STKLNK  Abs   62447 #0F3EF -   557   524   585
 STRHDR  Abs   62543 #0F44F -   590   529
=STRING  Abs   62364 #0F39C -   530
=STRRCL  Abs   62476 #0F40C -   567   552
 STRVCT  Abs   62434 #0F3E2 -   553   544
 STSAVE  Ext                -   923   960
 SYSEN   Ext                -   178
=TRMNTR  Abs   61917 #0F1DD -   176
 VECTLK  Abs   62320 #0F370 -   515   542
 VECTOR  Abs   62206 #0F2FE -   476   515
 ZERO    Abs   62163 #0F2D3 -   459   430
=csrc5   Abs   62664 #0F4C8 -   697
```

Input Parameters

  Source file name is AB&EXP::MS

  Listing file name is AB/EXP:TI:ML::-1

  Object file name is AB%EXP:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                      TITLE  Assignment Statement Controller<831216.1822>
 2 0F5DA                ABS    #0F5DA
 3                *       A    BBBB    &       A     SSS   N   N
 4                *      A A   B   B  & &     A A   S   S  N   N
 5                *     A   A  B   B  & &    A   A  S      NN  N
 6                *     A   A  BBBB    &     A   A   SSS   N N N
 7                *     AAAAA  B   B  & & &  AAAAA      S  N  NN
 8                *     A   A  B   B  &  &   A   A  S   S  N   N
 9                *     A   A  BBBB    && &  A   A   SSS   N   N
10                ************************************************************
11                      RDSYMB TI%EQU::MS
12                ************************************************************
13                ************************************************************
14                **
15                ** Name:(S) ASNMNT  -  Perform Variable Assignment
16                ** Name:    ASNSTO  -  Perform Variable Assignment
17                **
18                ** Category:   STEXEC
19                **
20                ** Purpose:
21                **      Evaluate expression and assign it to a variable.
22                **      ASNMNT evaluates (i.e., locates) destination variable.
23                **      ASNSTO does not (and requires proper entry conditions
24                **        for DEST subroutine).
25                **
26                ** Entry:
27                **              ASNMNT - D0 @ Destination Variable token.
28                **              ASNSTO - D0 1 byte before start of expression,
29                **                       Entry conditions for DEST.
30                **
31                **              S15 set if trace is desired.
32                **
33                ** Exit:        Top 16 nibbles of Mathstack in A,
34                **              D0 @ end of Statement,
35                **              D1 @ top of Mathstack.
36                **
37                ** Calls:       DEST, EXPEX-, SVTRC.  All STORE calls (below)
38                **
39                ** Uses:        Everything.
40                **
41                ** Stk lvls:    6
42                **
43                ** History:     SA and SC
44                **
45                **    Date      Programmer             Modification
46                **  --------    ----------   ---------------------------------
47                **  05/26/82    SA           Personnel change
48                **
49                ************************************************************
50                ************************************************************
51
52 0F5DA 8C00 =NOMEM  GOLONG =MEMERR        INSUFFICIENT MEMORY
         00
53 0F5E0 86F  =ASNMNT ?ST=0  15             IN TRACE MODE ?
54 0F5E3 60           GOYES  ASNM10         IF NOT, GOTO ASNM10
```

```
 55 0F5E5 7C44          GOSUB  SVTRC          SAVE STATEMENT ADDR IN S-R1-2
 56              *
 57              * Assignment calls Expression Execute at EXPEXC
 58              *  (Builds from TFORN)
 59              *
 60
 61 0F5E9 7000  ASNM10 GOSUB  =EXPEX-         LOCATE VARIABLE
 62 0F5ED 7FB1 =ASNSTO GOSUB  DEST            SAVE DESTINATION INFORMATION
 63 0F5F1 161           D0=D0+ 2
 64 0F5F4 7000          GOSUB  =EXPEX-        COMPUTE RESULT
 65              ****************************************************************
 66              ****************************************************************
 67              **
 68              ** Name:(S) STORE   -  Store From Stack To Variable
 69              **
 70              ** Category:   STEXEC
 71              **
 72              ** Purpose:    Store number or string in known register.
 73              **
 74              ** Entry:      Exit conditions of DEST
 75              **             D1 = (MTHSTK) = true top of Mathstack,
 76              **             Top 16 nibbles of Mathstack in A.
 77              **             Statement scratch has information set up by
 78              **               DEST.
 79              **             S-R1-2= Address points at the variable name
 80              **               This address is for TRACE to decompile the
 81              **               variable name. If the content of S-R1-2 is
 82              **               zero, the assigment will not be traced.
 83              **
 84              ** Exit:       Preserves D0,
 85              **             D1 @ top of Mathstack,
 86              **             R3 contains value stored in variable location
 87              **              (as opposed to the value in the RES register)
 88              **
 89              ** Calls:      CPOLL, Create, INTGR, RESTOR, SHRT, STRASN.
 90              **
 91              ** Uses:       Everything.
 92              **
 93              ** Stk lvls:   5      (TRACEA and CREATE)
 94              **
 95              ** History:
 96              **
 97              **    Date      Programmer      Modifications
 98              **  --------    ----------      -------------
 99              **  03/14/83    SW              R3 contains value stored
100              **
101              ****************************************************************
102              ****************************************************************
103              *
104              * NOTE: Above code falls into this code
105              *
106 0F5F8 20   =STORE  P=     0
107 0F5FA 136          CDOEX
108 0F5FD 06           RSTK=C                  SAVE PC ON HARDWARE STACK
109 0F5FF 1B00         D0=(5) =S-R0-0
```

```
          000
  110 0F606 7AF1          GOSUB  Create         CREATE VARIABLE IF NECESSARY
  111 0F60A 4FC           GOC    NOMEM          IF OUT OF MEMORY, GOTO NOMEM
  112 0F60D 309           LCHEX  9
  113 0F610 986           ?A>C   P              REAL RESULT?
  114 0F613 03            GOYES  STORE4         IF NOT, GOTO STORE4
  115 0F615 A4E           C=C-1  S              REAL DESTINATION?
  116 0F618 506           GONC   STORE5         IF NOT, GOTO STORE5
  117 0F61B 1507          DAT0=A W              STORE REAL RES IN REAL DEST
  118 0F61F 103           R3=A
  119 0F622 1B00  STORE1  DO=(5) =RESREG
          000
  120 0F629 1507          DAT0=A W              STORE RESULT IN RESULT REGISTER
  121 0F62D 04    STORE2  SETHEX
  122 0F62F 86F           ?ST=0  15             TRACE MODE?
  123 0F632 60            GOYES  TRCRTN         IF NOT, GOTO TRCRTN
  124 0F634 6000          GOTO   =TRACEA        PERFORM TRACE
  125 0F638 75F2 =TRCRTN  GOSUB  RESTOR         RESTORE TOP OF STACK TO A
  126 0F63C 07            C=RSTK                RESTORE PC TO D0
  127 0F63E 134           D0=C
  128 0F641 03            RTNCC                 RETURN CARRY CLEAR
  129
  130 0F643 B04   STORE4  A=A+1  P
  131 0F646 A64          A=A+A  B
  132 0F649 96C          ?A#0   B               STRING RESULT?
  133 0F64C 15           GOYES  STORE6          IF NOT, GOTO STORE6
  134 0F64E B46          C=C+1  S
  135 0F651 B46          C=C+1  S
  136 0F654 B46          C=C+1  S               STRING DESTINATION?
  137 0F657 514          GONC   TYPLK1          IF NOT, GOTO TYPERR
  138 0F65A 7550         GOSUB  STRASN          STORE STRING IN DESTINATION
  139 0F65E 5EC          GONC   STORE2          STORAGE OK THEN GOTO STORE2
  140 0F661 6DE2         GOTO   ERROR           GOTO ERROR HANDLER
  141
  142 0F665 7303  R5STO  GOSUB  SHRT            STORE REAL RES IN SHORT DEST
  143 0F669 1537  AGAIN  A=DAT1 W               REREAD RESULT
  144 0F66D 64BF         GOTO   STORE1          GOTO STORE1
  145 0F671 7623  I5STO  GOSUB  INTGR           STORE REAL RES IN INTEGER DEST
  146 0F675 63FF         GOTO   AGAIN           GOTO AGAIN
  147 0F679 AC5   STORE5 B=C    S               PREPARE FOR COMPLEX
  148 0F67C A4E          C=C-1  S               SHORT DESTINATION?
  149 0F67F 45E          GOC    R5STO           IF SO, GOTO R5STO
  150 0F682 A4E          C=C-1  S               INTEGER DESTINATION?
  151 0F685 4BE          GOC    I5STO           IF SO, GOTO I5STO
  152 0F688 3196         LC(2)  =cR->C
  153 0F68C 8E00  POLL   GOSUBL =CPOLL          POLL FOR COMPLEX
          00
  154 0F692 560          GONC   TYPLK1          IF NOT HANDLED, GOTO TYPERR
  155 0F695 679F         GOTO   STORE2          GOTO STORE2
  156             *      A=0    W               ZERO IMAGINARY PART
  157             *      B=B+1  S               COMPLEX SHORT DESTINATION?
  158             *      GOC    C5STO           IF SO, GOTO C5STO
  159             *      B=B+1  S               COMPLEX DESTINATION?
  160             *      GOC    C12STO          IF SO, GOTO C12STO
  161 0F699 61B2  TYPLK1 GOTO   TYPERR          GOTO TYPERR
```

```
162
163            *STORE6 LCHEX  E
164            *       ?A#C   B              COMPLEX RESULT?
165            *       GOYES  TYPERR         IF NOT, GOTO TYPERR
166 0F69D 3186  STORE6 LC(2)  =cC->C
167 0F6A1 6AEF         GOTO   POLL
168            *       D1=D1+ 2
169            *       A=DAT1 W              READ IMAGINARY PART
170            *       D1=D1+ 16
171            *       C=C+1  S              COMPLEX SHORT DESTINATION?
172            *       GOC    C5STO          IF SO, GOTO C5STO
173            *C5STO  GOSUB  SHRT           WRITE IMAGINARY PART TO DEST
174            *       D0=D0+ 3
175            *       A=DAT1 W              READ REAL PART
176            *       GOSUB  SHRT           WRITE REAL PART TO DESTINATION
177            *       A=DAT1 W
178            *       GOTO   STORE7
179            *C12STO DAT0=A W
180            *       D0=D0+ 16
181            *       A=DAT1 W
182            *       DAT0=A W
183            *STORE7 D0=(5) (=RESREG)+18
184            *       DAT0=A W
185            *       D1=D1- 2
186            *       A=DAT1 B
187            *       D0=D0- 2
188            *       DAT0=A B
189            *       D1=D1- 16
190            *STORE8 A=DAT1 W
191            *       GOTO   STORE1         GOTO STORE1
192            ********************************************************************
193            ********************************************************************
194            **
195            ** Name:(S) STRASN  -  String Assignment
196            **
197            ** Category:   VARMGT
198            **
199            ** Purpose: Store a string from stack to a string variable
200            **
201            ** Entry:
202            **      D1 = Stack pointer
203            **      A  = String header from stack ( A=DAT1 W)
204            **      S-R0-0 = Destination address ( @ String length)
205            **             = 00000 if hokey destination.
206            **
207            ** Exit:
208            **      P = 0
209            **      Carry clear => No error
210            **          Carry set => String too long
211            **
212            ** Calls:      MOVED3, MOVEU3
213            **
214            ** Uses:       A,B,C,D
215            **
216            ** Stk lvls:   2
```

```
                **
                ** History :
                **
                ** Date        Programmer     Modification
                ** ----------  -----------  ------------------------------
                ** 6/17/82     SC           straight line code=> subroutine
                **
                ********************************************************************
                ********************************************************************
                *
OF6A5 07     TOOLNG C=RSTK                 TRASH STUFF ON STACK
OF6A7 07            C=RSTK
OF6A9 D2            C=0    A
OF6AB 3100          LC(2)  =eSTROV
OF6AF 02            RTNSC                 RETURN CARRY SET
OF6B1 03     HOKEY  RTNCC                 RETURN CARRY CLEAR
OF6B3 BF4   =STRASN ASR    W
OF6B6 BF4           ASR    W
OF6B9 1B00          D0=(5) =S-R0-0
      000
OF6C0 146           C=DAT0 A
OF6C3 8AA           ?C=0   A              HOKEY DESTINATION?
OF6C6 BE            GOYES  HOKEY          IF SO, GOTO HOKEY
OF6C8 137           CD1EX
OF6CB 06            RSTK=C                STACK POINTER ON HARDWARE STACK
OF6CD D2            C=0    A
OF6CF 15F3          C=DAT1 4
OF6D3 C6            C=C+C  A
OF6D5 D5            B=C    A              CURRENT DESTINATION LENGTH IN B
OF6D7 C0            A=A+B  A
OF6D9 164           D0=D0+ 5
OF6DC 146           C=DAT0 A
OF6DF AF3           D=0    W
OF6E2 D7            D=C    A              START POSITION IN D
OF6E4 CA            A=A+C  A
OF6E6 164           D0=D0+ 5
OF6E9 146           C=DAT0 A              END POSITION IN C
OF6EC 8B1           ?B>C   A              END AFTER CURRENT END?
OF6EF 40            GOYES  $TORE1         IF SO, SET END TO CURRENT END
OF6F1 D9            C=B    A
OF6F3 EA     $TORE1 A=A-C  A              NEW OVERALL LENGTH IN A
OF6F5 25            P=     5
OF6F7 A80           A=0    P
OF6FA 20            P=     0
OF6FC 81C           ASRB
OF6FF 06            RSTK=C
OF701 16A           D0=D0+ 11
OF704 146           C=DAT0 A              STRING ALLOTMENT IN C
OF707 8B6           ?A>C   A              STORAGE OVERFLOW?
OF70A B9            GOYES  TOOLNG         IF SO, GOTO TOOLNG
OF70C 1593          DAT1=A 4              STORE NEW STRING LENGTH
OF710 C4            A=A+A  A
OF712 07            C=RSTK
OF714 173           D1=D1+ 4
OF717 8BC           ?A<=B  A              NEW STRING SHORTER THAN OLD?
```

```
271 0F71A 65          GOYES  $TORE5         IF SO, GOTO $TORE5
272 0F71C 137         CD1EX
273 0F71F CA          A=A+C  A
274 0F721 131         D1=A                  NEW START OF STRING POS'N IN D1
275 0F724 C9          C=B+C  A
276 0F726 134         D0=C                  CURR START OF STRING POS'N IN D0
277 0F729 DB          C=D    A
278 0F72B DD          BCEX   A
279 0F72D E3          D=D-C  A              NUMBER OF BLANKS TO FILL IN D
280 0F72F 560         GONC   $TORE2         LENGTH OF LEADER STRING IN C
281 0F732 D3          D=0    A
282 0F734 D9          C=B    A
283 0F736 81F  $TORE2 DSRB
284 0F739 8E00        GOSUBL =MOved3        MOVE STRING TO NEW POSITION
          00
285 0F73F 3102        LCASC  \ \
286 0F743 580         GONC   $TORE4         UNCONDITIONAL GOTO $TORE4
287 0F746 1C1  $TORE3 D1=D1- 2
288 0F749 14D         DAT1=C B              WRITE BLANK
289 0F74C CF   $TORE4 D=D-1  A              MORE BLANKS TO FILL?
290 0F74E 57F         GONC   $TORE3         IF SO, REPEAT $TORE3
291 0F751 07          C=RSTK
292 0F753 134         D0=C
293 0F756 161         D0=D0+ 2
294 0F759 142         A=DAT0 A
295 0F75C DE          ACEX   A              LENGTH OF RESULT IN C
296 0F75E CA          A=A+C  A
297 0F760 130         D0=A                  START OF RESULT POSITION IN D0
298 0F763 16F         D0=D0+ 16
299 0F766 8E00        GOSUBL =MOved3        MOVE RESULT INTO DESTINATION
          00
300 0F76C 6430        GOTO   $TORE7
301 0F770 133  $TORE5 AD1EX
302 0F773 EE          C=A-C  A
303 0F775 C9          C=B+C  A
304 0F777 137         CD1EX                 TRAILER START POSITION IN D1
305 0F77A C8          B=A+B  A
306 0F77C DB          C=D    A
307 0F77E E1          B=B-C  A              HEADER END POSITION IN B
308 0F780 07          C=RSTK
309 0F782 134         D0=C
310 0F785 161         D0=D0+ 2
311 0F788 146         C=DAT0 A              RESULT LENGTH IN C
312 0F78B 16D         D0=D0+ 14
313 0F78E 8E00        GOSUBL =Moveu3        MOVE RESULT INTO DESTINATION
          00
314 0F794 D9          C=B    A
315 0F796 134         D0=C
316 0F799 DB          C=D    A              LEADER LENGTH IN C
317 0F79B 8E00        GOSUBL =Moveu3        CLOSE LEADER-RESULT GAP
          00
318 0F7A1 1F00 $TORE7 D1=(5) =MTHSTK
          000
319 0F7A8 147         C=DAT1 A
320 0F7AB 135         D1=C
```

```
 321 0F7AE 03           RTNCC
 322              ********************************************************************
 323              ********************************************************************
 324              **
 325              ** Name:(S) DEST    -  Save Variable Destination Info
 326              **
 327              ** Category:   VARMGT
 328              **
 329              ** Purpose:
 330              **      Save variable destination information for use by STORE
 331              **      subroutine.
 332              **
 333              ** Entry:
 334              **      B      =  Exit condition from EXPEXC (see note below)
 335              **      D1     =  Exit condition from EXPEXC (see note below)
 336              **      F-R1-0 =  Exit condition from EXPEXC (see note below)
 337              **      F-R1-3 =  Exit condition from EXPEXC (see note below)
 338              **
 339              ** Exit:
 340              **      P=0.
 341              **      Following information has been stored:
 342              **        S-R0-1 = First substring parameter.
 343              **        S-R0-2 = Second substring parameter.
 344              **        S-R0-3 = Variable type.
 345              **        S-R1-0 = Array element number.
 346              **        S-R1-1 = Maximum string length.
 347              **        S-R1-3 = Subscript count.
 348              **
 349              ** Calls:      None.
 350              **
 351              ** Uses.......
 352              **             D1,C.
 353              **
 354              ** Stk lvls:   0
 355              **
 356              ** NOTE:
 357              **      Whenever EXPEXC evaluates a variable (simple or array
 358              **      element), it leaves destination information about that
 359              **      variable in B[W] and function scratch.  This routine
 360              **      puts that information in statement scratch, where it is
 361              **      safe from further abuse during expression execute, and
 362              **      can be subsequently accessed for a store operation.
 363              **
 364              **      In computing the destination information, the recall
 365              **      code sets up information about the variable's address,
 366              **      substring parameters, type, array register number,
 367              **      maximum string length and subscript count.  If the
 368              **      variable does not exist, that fact is somehow encoded
 369              **      into this information and the variable will be created
 370              **      in the store subroutine.
 371              **
 372              ** Detail:
 373              **      Typically called after EXPEXC, which left information
 374              **      around about the location of the last variable
 375              **      evaluated (if evaluating a variable was the last thing
```

```
376                **      done).  Typical use is in variable assignment:
377                **        EXPEXC (evaluate destination variable).
378                **        DEST (save destination information for STORE).
379                **        EXPEXC (evaluate expression).
380                **        STORE (store result in destination variable).
381                **
382                ** History:
383                **
384                **    Date     Programmer              Modification
385                **  --------   ----------   ---------------------------------
386                **             SA           Wrote
387                **  10/13/83   NM           Attempted to document
388                **
389                ****************************************************************
390                ****************************************************************
391 0F7B0 17B   =DEST   D1=D1+ 12
392 0F7B3 D2           C=0    A
393 0F7B5 15F3         C=DAT1 4
394 0F7B9 1F00         D1=(5) =S-R1-1        STORE DESTINATION INFORMATION:
          000
395 0F7C0 145          DAT1=C A              S-R0-0 = VAR ADDRESS OR LABEL
396 0F7C3 AF9          C=B    W              S-R0-1 = FIRST SUBSTR PARAMETER
397 0F7C6 1F00         D1=(5) =S-R0-0        S-R0-2 = SECOND SUBSTR PARAMETER
          000
398 0F7CD 1557         DAT1=C W              S-R0-3 = VARIABLE TYPE
399 0F7D1 1E00         D1=(4) =F-R1-0        S-R1-0 = ARRAY REGISTER NUMBER
          00
400 0F7D7 1577         C=DAT1 W              S-R1-1 = MAXIMUM STRING LENGTH
401 0F7DB 1E00         D1=(4) =S-R1-0        S-R1-2 =
          00
402 0F7E1 145          DAT1=C A              S-R1-3 = SUBSCRIPT COUNT
403 0F7E4 17E          D1=D1+ 15
404 0F7E7 1554         DAT1=C S
405 0F7EB 01           RTN                   RETURN
406                ****************************************************************
407                *BUGFIX 9605 ON 830803 -SA
408                *This table renamed =B9605A, and moved to a Romfix area.
409                *
410                *DCGLOP NIBHEX 0B0            NUMERIC, BASE 0, DIM 1
411                *       NIBHEX 097            NUMERIC, BASE 0, DIM 2
412                *       NIBHEX 0A0            NUMERIC, BASE 1, DIM 1
413                *       NIBHEX 046            NUMERIC, BASE 1, DIM 2
414                *       NIBHEX CE2            STRING,  BASE 0, DIM 1
415                *       NIBHEX 440            STRING,  BASE 0, SIMPLE
416                *       NIBHEX 8A2            STRING,  BASE 1, DIM 1
417                *       NIBHEX 440            STRING,  BASE 1, SIMPLE
418                ************************************
419 0F7ED 0           CON(1) =FIXSPC        Log that freespace is available
420 0F7EE           BSS    18
421                ************************************
422 0F800 6A41   TYPLK2 GOTO   TYPERR
423
424                ****************************************************************
425                ****************************************************************
426                **
```

```
427             ** Name:    Create  -  Create Variables
428             ** Name:    NEWVAR  -  Create Variables
429             ** Name:    DFLTCR  -  Create Variables
430             **
431             ** Category:   VARMGT
432             **
433             ** Purpose:
434             **      Create is a local label.
435             **      NEWVAR and DFLTCR are entry points in create code
436             **        to create new variables.
437             **
438             ** Entry:
439             **      Unclear.
440             **
441             ** Exit:
442             **      Unclear.
443             **
444             ** Calls:      AJDEST, BASE, CR-ARR, CR-VAR, chkspc.
445             **
446             ** Uses.......
447             **             Unclear.
448             **
449             ** Stk lvls:   Unclear.
450             **
451             ** Detail:
452             **      Create uses information saved in the statement scratch
453             **      (among other places?) to create variables.  It is
454             **      recommended that you not try to use this routine to
455             **      create an array.  Instead, follow the instructions in
456             **      the IDS volume 1, which tell you how to point at a
457             **      tokenized DIM statement and execute it.
458             **
459             ** History:
460             **
461             **    Date     Programmer             Modification
462             **  --------   ----------   ---------------------------------
463             **             SA           Wrote
464             **  10/13/83   NM           Attempted to document
465             **
466             ********************************************************************
467             ********************************************************************
468 0F804 34E0  Create LCHEX  0100E
        010
469 0F80B D7            D=C    A
470 0F80D 1567          C=DATO W              READ TYPE & ADDRESS
471 0F811 8B3           ?C<D   A              DOES VARIABLE EXIST?
472 0F814 70            GOYES  CR20           IF NOT, GOTO CR20
473 0F816 134   CR10    DO=C                  DO POINTS AT VARIABLE
474 0F819 03            RTNCC                 RETURN CARRY CLEAR
475
476 0F81B 8AA   CR20    ?C=0   A              BUGFIX 9605 ON 830803 -SA
477 0F81E 8F            GOYES  CR10
478 0F820 DE            ACEX   A
479 0F822 963           ?C=D   B              COMPLEX NUMBER ON STACK?
480 0F825 BD            GOYES  TYPLK2         IF SO, GOTO TYPERR
```

```
481 0F827 DE            ACEX   A
482 0F829 8AA  =NEWVAR ?C=0    A              HOKEY ARRAY ELEMENT?
483 0F82C AE            GOYES  CR10            IF SO, GO AHEAD & WRITE TO 00000
484 0F82E D5            B=C    A              SAVE NAME IN B(A)
485 0F830 A46           C=C+C  S              STRING OR ARRAY?
486 0F833 432           GOC    DFLTCR         IF SO, GOTO DFLTCR
487 0F836 D2            C=0    A
488 0F838 7801          GOSUB  chkspc         ENOUGH ROOM?
489 0F83C 400           RTNC                  IF NOT, RETURN CARRY SET
490 0F83F 8E00          GOSUBL =CR-VAR        CREATE VARIABLE
          00
491 0F845 3100          LC(2)  =eVCNTX
492 0F849 490           GOC    UUURP          IF IT EXISTS, GOTO ERROR
493 0F84C 136           CDOEX
494 0F84F 64D0          GOTO   GETBAK         GOTO GETBAK
495 0F853 6000  UUURP   GOTO   =MfErr
496
497 0F857 841   =DFLTCR ST=0   1              DENOTE NUMERIC
498 0F85A 3AC0          NIBHEX 3AC00          LOAD NUMERIC DOPE VECTOR SKELETON
          0
499 0F85F A000 =DIM10   NIBHEX A000A000
          A000
500 0F867 1B00          D0=(5) =S-R1-3
          000
501 0F86E 21            P=     1
502 0F870 1560          C=DAT0 P              READ SUBSCRIPT COUNT
503 0F874 1F00          D1=(5) =B9605A
          000
504 0F87B A46           C=C+C  S              NUMERIC ARRAY?
505 0F87E 5C2           GONC   DC10           IF SO, GOTO DC10
506 0F881 851           ST=1   1              DENOTE STRING
507 0F884 17B           D1=D1+ 12
508 0F887 D2            C=0    A
509 0F889 302           LCHEX  2              CONSTRUCT DEFAULT MAXIMUM LENGTH
510 0F88C 189           D0=D0- 10
511 0F88F 142           A=DAT0 A              READ ARRAY REGISTER NUMBER
512 0F892 144           DAT0=C A              WRITE MAXIMUM STRING LENGTH
513 0F895 184           D0=D0- 5
514 0F898 140           DAT0=A A              WRITE ARRAY REGISTER NUMBER
515 0F89B CE            C=C-1  A              LOAD LOWER END OF STRING D.V.
516 0F89D 3310          LCHEX  2001
          02
517 0F8A3 94A           ?C=0   S              STRING ARRAY?
518 0F8A6 50            GOYES  DC10           IF SO, GOTO DC10
519 0F8A8 300           LCHEX  0              INDICATE SIMPLE STRING
520 0F8AB 74A0  DC10    GOSUB  BASE           GET CURRENT BASE OPTION
521 0F8AF 109           R1=C                  SAVE SKELETON IN R1
522 0F8B2 550           GONC   DC20           IF BASE = 0 THEN GOTO DC20
523 0F8B5 175           D1=D1+ 6
524 0F8B8 A0E   DC20    C=C-1  P
525 0F8BB 90A           ?C=0   P              DIMENSION COUNT = 1 ?
526 0F8BE 50            GOYES  DC30           IF SO, GOTO DC30
527 0F8C0 172           D1=D1+ 3
528 0F8C3 AF2   DC30    C=0    W
529 0F8C6 1573          C=DAT1 X              READ SPACE REQUIREMENT
```

```
530 0F8CA 108          RO=C
531 0F8CD 20           P=     0
532 0F8CF 7170         GOSUB  chkspc          ENOUGH ROOM?
533 0F8D3 400          RTNC                  IF NOT, RETURN CARRY SET
534 0F8D6 8E00         GOSUBL =CR-VAR        CREATE VARIABLE
          00
535 0F8DC 3100         LC(2)  =eVCNTX
536 0F8E0 4E6          GOC    ERROR          IF IT EXISTS, GOTO ERROR
537            *       GONC   DC40           IF NEW, GOTO DC50
538            *       A=DAT0 B
539            *       LCHEX  9
540            *       ?A>C   P              SIMPLE REAL VARIABLE?
541            *       GOYES  TYPERR         IF NOT, GOTO TYPERR
542 0F8E3 8E00  DC40   GOSUBL =CR-ARR        CREATE ARRAY
          00
543 0F8E9 8E00         GOSUBL =AJDEST        ADJUST DESTINATION ADDRESS
          00
544 0F8EF 1F00         D1=(5) =S-R1-0
          000
545 0F8F6 143          A=DAT1 A              READ REGISTER NUMBER
546 0F8F9 D6           C=A    A
547 0F8FB F0           ASL    A              MULTIPLY REGNUM BY 16
548 0F8FD 2F           P=     15
549 0F8FF 300          LCHEX  0              LOAD REAL TYPE INDICATOR
550 0F902 861          ?ST=0  1              NUMERIC VARIABLE?
551 0F905 B0           GOYES  DC50           IF SO, GOTO DC50
552 0F907 CA           A=A+C  A              MULTIPLY REGNUM BY 68
553 0F909 C4           A=A+A  A
554 0F90B C4           A=A+A  A
555 0F90D 30D          LCHEX  D              LOAD STRING TYPE INDICATOR
556 0F910 20    DC50   P=     0
557 0F912 D9           C=B    A
558 0F914 134          DO=C
559 0F917 16A          DO=DO+ 11
560 0F91A 146          C=DAT0 A              READ ARRAY RELATIVE OFFSET
561 0F91D E2           C=C-A  A              COMPUTE REGISTER OFFSET
562 0F91F 132          ADOEX
563 0F922 EE           C=A-C  A              COMPUTE REGISTER ADDRESS
564 0F924 1F00  GETBAK D1=(5) =S-RO-0
          000
565 0F92B 145          DAT1=C A              STORE ADDRESS IN S-RO-0
566 0F92E 134          DO=C                  LEAVE ADDRESS IN DO
567 0F931 1F00  RESTOR D1=(5) =MTHSTK
          000
568 0F938 143          A=DAT1 A
569 0F93B 131          D1=A                  RESTORE MATH STACK POINTER
570 0F93E 1537         A=DAT1 W              RESTORE TOP OF MATH STACK
571 0F942 03           RTNCC                 RETURN CARRY CLEAR
572 0F944 8D00 =chkspc GOVLNG =CHKSPC
          000
573 0F94B 3100  TYPERR LC(2)  =eDATTY
574 0F94F 6000  ERROR  GOTO   =MfErr
575            ********************************************************************
576            ********************************************************************
577            **
```

```
578                ** Name:(S) BASE    -  Determine Option Base
579                **
580                ** Category:   VARMGT
581                **
582                ** Purpose:
583                **      Determine whether we are in option base 0 or 1.
584                **
585                ** Entry:
586                **      HEX mode.
587                **
588                ** Exit:
589                **      If carry set:
590                **        We are in option base 1.  C[XS]=1.
591                **      If carry clear:
592                **        We are in option base 0.  C[XS]=0.
593                **
594                ** Calls:      None.
595                **
596                ** Uses.......
597                **             D0,C[XS].
598                **
599                ** Stk lvls:   0
600                **
601                ** History:
602                **
603                **    Date     Programmer              Modification
604                **  --------   ----------   ------------------------------------
605                **             SA           Wrote
606                **  10/13/83   NM           Attempted to document
607                **
608                ********************************************************************
609                ********************************************************************
610 0F953 1B00 =BASE   D0=(5) (=SYSFLG)+3    GET CURRENT BASE OPTION
          000
611 0F95A 1562         C=DAT0 XS
612 0F95E A26          C=C+C  XS
613 0F961 AA2          C=0    XS
614 0F964 500          RTNNC                  IF ZERO, RETURN CARRY CLEAR
615 0F967 B26          C=C+1  XS
616 0F96A 02           RTNSC                  IF ONE, RETURN CARRY SET
617                ********************************************************************
618                ********************************************************************
619                **
620                ** Name:(S) SHRT    -  Store Into Short Variable
621                **
622                ** Category:   VARMGT
623                **
624                ** Purpose:
625                **      Store a number into a short variable, with IEEE
626                **      rounding.
627                **
628                ** Entry:
629                **      12-digit form in A[W].
630                **      D0 pointing at variable storage location.
631                **
```

```
632            ** Exit:
633            **      R3 contains copy of number as stored.
634            **      DEC mode
635            **
636            ** Calls:      SPLITA, uRESNX.
637            **
638            ** Uses.......
639            **             D0,D1,A,B,C,D,R0,R3,S7-S11.
640            **
641            ** Stk lvls:   3
642            **
643            ** History:
644            **
645            **    Date     Programmer             Modification
646            **  --------   ----------   ----------------------------------
647            **             SA           Wrote
648            **  10/13/83   NM           Attempted to document
649            **
650            ********************************************************************
651            ********************************************************************
652 0F96C 05   =SHRT   SETDEC
653
654
655            *  ovf/unf check   *         NOTE: 9.99996E499 ovflws  &
656                                                 0.0012345E-499 unflws.
657 0F96E 8E00         GOSUBL =SPLITA
           00
658 0F974 821          XM=0
659 0F977 822          SB=0
660 0F97A 29           P=     9
661 0F97C 8E00         GOSUBL =uRESNX        IEEE RND AT DIG 5, ck traps, set
           00
662                                          flags, disp messages.
663 0F982 10B          R3=C
664                                           (uses R3 & s7...s11 also)
665 0F985 DA           A=C    A              A(X)=exp of ans, C=sgn&mant&expon
666            *----------------------*
667
668
669 0F987 189          D0=D0- 10
670 0F98A 15E9         C=DAT0 10             READ JUNK
671 0F98E 1547         DAT0=C W              WRITE MANTISSA, SIGN, & JUNK
672 0F992 16F          D0=D0+ 16
673 0F995 1503         DAT0=A X              WRITE EXPONENT
674 0F999 01           RTN                   RETURN
675            ********************************************************************
676            ********************************************************************
677            **
678            ** Name:(S) INTGR   -  Store Into An Integer Variable
679            **
680            ** Category:   VARMGT
681            **
682            ** Purpose:
683            **      Store a number into an integer variable.
684            **
```

```
685             ** Entry:
686             **      Number in 12-digit floating-point form in A.
687             **
688             ** Exit:
689             **      P       =  0
690             **
691             ** Calls:       IF12A, OVFL, RND-12, SIGCHK, uRESXT.
692             **
693             ** Uses.......
694             **              A,B,C,D,D0,D1,R0,R3,S7-S11.
695             **
696             ** Stk lvls:    3
697             **
698             ** Detail:
699             **      Handles overflow according to IEEE trap settings.
700             **
701             ** History:
702             **
703             **    Date      Programmer              Modification
704             **  --------    ----------    ----------------------------------
705             **              SA            Wrote
706             **  10/13/83    NM            Attempted to document
707             **
708             ********************************************************************
709             ********************************************************************
710 0F99B 05    =INTGR  SETDEC                CAUTION - PACKED CODE
711
712             *  Check for Sig NaNs -- unless we encode INTEGER SigNaNs, *
713 0F99D 8E00         GOSUBL =SIGCHK             then remove this check.
        00
714             *
715
716 0F9A3 8E00         GOSUBL =IF12A          FIND DECIMAL POINT
        00
717 0F9A9 AD4          A=B    M
718 0F9AC 8F00         GOSBVL =RND-12         ROUND AT DECIMAL POINT
        000
719 0F9B3 103          R3=A
720 0F9B6 04           SETHEX
721 0F9B8 AF2          C=0    W
722 0F9BB 2F           P=     15
723 0F9BD 3185         LCHEX  58
724 0F9C1 20           P=     0
725 0F9C3 0E46         A=A&C  S              TURN OFF LOW BIT OF SIGN DIGIT
726 0F9C7 9B2          ?A<C   X              LEGAL INTEGER RANGE?
727 0F9CA 45           GOYES  INTGR3         IF NOT, GOTO INTGR3
728 0F9CC 3200         LCHEX  500
        5
729 0F9D1 9BE          ?A>=C  X              0<= expon <= 4 ?
730 0F9D4 B2           GOYES  INTGR0         IF SO, GOTO INTGR0
731
732
733
734             * Handle Overflow  *
735 0F9D6 05           SETDEC
```

```
    736 0F9D8 840          ST=0   =sINFRD
    737             *      (P=0 from above)
    738 0F9DB 8E00         GOSUBL =OVFL          B(A)=eOVFLW,P=OVP,C=+-9.99..9F00
              00
    739 0F9E1 8E00         GOSUBL =uRESXT        ck. traps, set flags, disp msg.
              00
    740                                          C:=+-(Inf or 9.99..9e499) (traps)
    741 0F9E7 10B          R3=C
    742 0F9EA 04           SETHEX
    743 0F9EC AFA          A=C    W              A=+-(Inf or maxreal)
    744 0F9EF 20           P=     0
    745 0F9F1 306          LCHEX  6
    746 0F9F4 968          ?A=0   B              A=+-Inf ?
    747 0F9F7 A2           GOYES  INTGR4         rtn integer Inf.
    748 0F9F9 304          LCHEX  4
    749 0F9FC 542          GONC   INTGR4         "goto", A=+-99999.
    750             * -----------------*
    751
    752
    753
    754 0F9FF 3200 INTGR0  LCHEX  F00
              F
    755 0FA04 9B2          ?A<C   X              SMALL NUMBER?
    756 0FA07 11           GOYES  INTGR2         IF SO, GOTO INTGR2 (WRITE ZERO)
    757 0FA09 307          LCHEX  7
    758 0FA0C A0C          A=A-1  P              NAN?
    759 0FA0F 511          GONC   INTGR4         IF SO, GOTO INTGR4
    760 0FA12 306   INTGR1 LCHEX  6              CREATE INTEGER INFINITY:
    761 0FA15 A5E          C=C-1  M              FFFFFFFFFFFF006
    762 0FA18 A8A   INTGR2 A=C    P
    763 0FA1B ADA          A=C    M
    764 0FA1E A86   INTGR3 C=A    P
    765 0FA21 816   INTGR4 CSRC
    766 0FA24 0E4E         A=A!C  S              PACK SIGN & EXPONENT
    767 0FA28 189          DO=DO- 10
    768 0FA2B 15A9         A=DATO 10             READ JUNK
    769 0FA2F 1507         DATO=A W              WRITE INTEGER & JUNK
    770 0FA33 03           RTNCC                 RETURN CARRY CLEAR
    771             *
    772             *
    773             ***********************************************************************
    774             ***********************************************************************
    775             **
    776             ** Name:(S) SVTRC   -  Save Trace Information In Stmt Scratch
    777             **
    778             ** Category:   EXCUTL
    779             **
    780             ** Purpose:
    781             **      Save trace information in stmt scratch.
    782             **
    783             ** Entry:
    784             **      DO = trace information.
    785             **
    786             ** Exit:
    787             **      Copy of information in C[A].
```

```
788             **      Information saved in S-R1-2.
789             **
790             ** Calls:      None.
791             **
792             ** Uses.......
793             **             C[A]
794             **
795             ** Stk lvls:   0
796             **
797             ** History:
798             **
799             **    Date      Programmer              Modification
800             **  --------    ----------   ------------------------------------
801             **              SA           Wrote
802             **  11/01/83    NM           Attempted to document
803             **
804             ************************************************************************
805             ************************************************************************
806 0FA35 136   =SVTRC  CDOEX
807 0FA38 1B00 =SETTRC DO=(5) =S-R1-2
          000
808 0FA3F 144          DATO=C A
809 0FA42 134          DO=C
810 0FA45 01           RTN
811 0FA47              END
```

```
 $TORE1  Abs   63219 #0F6F3 -   256   254
 $TORE2  Abs   63286 #0F736 -   283   280
 $TORE3  Abs   63302 #0F746 -   287   290
 $TORE4  Abs   63308 #0F74C -   289   286
 $TORE5  Abs   63344 #0F770 -   301   271
 $TORE7  Abs   63393 #0F7A1 -   318   300
 AGAIN   Abs   63081 #0F669 -   143   146
 AJDEST  Ext                   543
 ASNM10  Abs   62953 #0F5E9 -    61    54
=ASNMNT  Abs   62944 #0F5E0 -    53
=ASNSTO  Abs   62957 #0F5ED -    62
 B9605A  Ext                   503
=BASE    Abs   63827 #0F953 -   610   520
 CHKSPC  Ext                   572
 CPOLL   Ext                   153
 CR-ARR  Ext                   542
 CR-VAR  Ext                   490   534
 CR10    Abs   63510 #0F816 -   473   477   483
 CR20    Abs   63515 #0F81B -   476   472
 Create  Abs   63492 #0F804 -   468   110
 DC10    Abs   63659 #0F8AB -   520   505   518
 DC20    Abs   63672 #0F8B8 -   524   522
 DC30    Abs   63683 #0F8C3 -   528   526
 DC40    Abs   63715 #0F8E3 -   542
 DC50    Abs   63760 #0F910 -   556   551
=DEST    Abs   63408 #0F7B0 -   391    62
=DFLTCR  Abs   63575 #0F857 -   497   486
=DIM10   Abs   63583 #0F85F -   499
 ERROR   Abs   63823 #0F94F -   574   140   536
 EXPEX-  Ext                    61    64
 F-R1-0  Ext                   399
 FIXSPC  Ext                   419
 GETBAK  Abs   63780 #0F924 -   564   494
 HOKEY   Abs   63153 #0F6B1 -   232   238
 I5STO   Abs   63089 #0F671 -   145   151
 IF12A   Ext                   716
=INTGR   Abs   63899 #0F99B -   710   145
 INTGR0  Abs   63999 #0F9FF -   754   730
 INTGR1  Abs   64018 #0FA12 -   760
 INTGR2  Abs   64024 #0FA18 -   762   756
 INTGR3  Abs   64030 #0FA1E -   764   727
 INTGR4  Abs   64033 #0FA21 -   765   747   749   759
 MEMERR  Ext                    52
 MOved3  Ext                   284   299
 MTHSTK  Ext                   318   567
 MfErr   Ext                   495   574
 Moveu3  Ext                   313   317
=NEWVAR  Abs   63529 #0F829 -   482
=NOMEM   Abs   62938 #0F5DA -    52   111
 OVFL    Ext                   738
 POLL    Abs   63116 #0F68C -   153   167
 R5STO   Abs   63077 #0F665 -   142   149
 RESREG  Ext                   119
 RESTOR  Abs   63793 #0F931 -   567   125
 RND-12  Ext                   718
```

```
 S-R0-0  Ext                  -   109   235   397   564
 S-R1-0  Ext                  -   401   544
 S-R1-1  Ext                  -   394
 S-R1-2  Ext                  -   807
 S-R1-3  Ext                  -   500
=SETTRC  Abs   64056 #0FA38  -   807
=SHRT    Abs   63852 #0F96C  -   652   142
 SIGCHK  Ext                  -   713
 SPLITA  Ext                  -   657
=STORE   Abs   62968 #0F5F8  -   106
 STORE1  Abs   63010 #0F622  -   119   144
 STORE2  Abs   63021 #0F62D  -   121   139   155
 STORE4  Abs   63043 #0F643  -   130   114
 STORE5  Abs   63097 #0F679  -   147   116
 STORE6  Abs   63133 #0F69D  -   166   133
=STRASN  Abs   63155 #0F6B3  -   233   138
=SVTRC   Abs   64053 #0FA35  -   806    55
 SYSFLG  Ext                  -   610
 TOOLNG  Abs   63141 #0F6A5  -   227   265
 TRACEA  Ext                  -   124
=TRCRTN  Abs   63032 #0F638  -   125   123
 TYPERR  Abs   63819 #0F94B  -   573   161   422
 TYPLK1  Abs   63129 #0F699  -   161   137   154
 TYPLK2  Abs   63488 #0F800  -   422   480
 UUURP   Abs   63571 #0F853  -   495   492
 cC->C   Abs     104 #00068  -    11   166
 cR->C   Abs     105 #00069  -    11   152
=chkspc  Abs   63812 #0F944  -   572   488   532
 eDATTY  Ext                  -   573
 eSTROV  Ext                  -   230
 eVCNTX  Ext                  -   491   535
 sINFRD  Ext                  -   736
 uRESNX  Ext                  -   661
 uRESXT  Ext                  -   739
```

Input Parameters

  Source file name is AB&ASN::MS

  Listing file name is AB/ASN:TI:ML::-1

  Object file name is AB%ASN:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS    CCC    &     TTTTT  RRRR    CCC
 2            *    S   S  C   C  & &      T    R   R  C   C
 3            *    S      C      & &      T    R   R  C
 4            *     SSS   C       &       T    RRRR   C
 5            *        S  C      & & &    T    R R    C
 6            *    S   S  C   C  &  &     T    R  R   C   C
 7            *     SSS    CCC    && &    T    R   R   CCC
 8            *
 9            *
10            *
11                    TITLE  TRACE Module <831216.1824>
12                    RDSYMB SB%RAM::MS
13                    RDSYMB TI%EQU::MS
14 0FA47              ABS    #0FA47
```

```
15                      EJECT
16                  ************************************************************
17                  ************************************************************
18                  **
19                  ** Name:    TRACE   -  TRACE Statement Execution
20                  **
21                  ** Category:   STEXEC
22                  **
23                  ** Purpose:  TRACE statement execution
24                  **
25                  ** Calls: D1=TRC
26                  **
27                  ** Detail:
28                  **   There is a nibble in the system RAM(TRACEM) to indicate
29                  **   the mode of trace:
30                  **   TRACEM = 0  - Trace off
31                  **            4  - Trace all variables
32                  **            2  - Trace flow
33                  **   S15 =1 if either in TRACE VARS or TRACE FLOW
34                  **   Trace mode are global status of the system.
35                  **   Trace flow and trace variables can be co-exist.
36                  **
37                  ************************************************************
38                  ************************************************************
39                  *
40                  SelTrc  EQU    1
41                  *
42                  *
43 0FA47 0000          REL(5) =TRACDC
         0
44 0FA4C 0000          REL(5) =TRACEP
         0
45 0FA51 AFA  =TRACE   A=C    W              A=C = 0
46 0FA54 35FE          LC(6)  =tVARS
         10B5
47 0FA5C 15A5          A=DAT0 6
48 0FA60 972           ?A=C   W
49 0FA63 21            GOYES  TRVARS         IF TRACE VARS, GOTO TRVARS
50 0FA65 84F           ST=0   15             ASSUME IS TRACE OFF
51 0FA68 35FE          LC(6)  =tFLOW
         1092
52 0FA70 976           ?A#C   W
53 0FA73 50            GOYES  TROFF
54 0FA75 85F   TRVARS  ST=1   15
55 0FA78 7DB4  TROFF   GOSUB  D1=TRC
56 0FA7C 540           GONC   TRFLOW         NO CARRY FOR TRACE FLOW
57 0FA7F C6            C=C+C  A
58 0FA81 0E0A  TRFLOW  C=A!C  P
59                  *
60 0FA85 87F           ?ST=1  15
61 0FA88 40            GOYES  TRAL20
62 0FA8A D2            C=0    A
63 0FA8C 1550  TRAL20  DAT1=C P
64 0FA90 8C00          GOLONG =NXTSTM
         00
```

```
65            **
66            ********************************************************************
67            ********************************************************************
68            **
69            ** Name:    TRACEA  -  Trace variable assignment
70            **
71            ** Category:   VARMGT
72            **
73            ** Purpose:  Trace variable assignment
74            **
75            ** Entry:
76            **     S13 = System run flag
77            **     S-R1-2 = Address of the variable token in the statement.
78            **     S-R1-2 = 0 to disable the TRACE VARS for one assignment.
79            **     STMTR0 & STMTR1 have not been altered since last time
80            **      called the DEST. STMTR0 & STMTR1 have the information
81            **      of the destination, such as variable type, array
82            **      subscript, etc.
83            **
84            ** Exit:
85            **          This routine is called by the STORE routine, so it
86            **          return to STORE thru a hardwire jump to =TRCRTN.
87            **
88            **
89            ** Calls:  EXPRDC,DRANGE,ADRS45,TRCLIN,DSBFCK,HEXDEC,LIN#AU
90            **         STR$SB, DSPCNA, POPMTH, AVE=D1, REVPOP, TRFM20, FNCK
91            **         D0=PCA, RCLNUM, D=AVME, D0=AVS, OUT2TC, D1=SR1
92            **         VARDC, IDIV, D1=TRC, BSTYCK, FCALC?, DATLEN.
93            **
94            ** Uses:       A-D, D0,D1, R0,R1,R2, S0,S3,S5,S8-11
95            *
96            ** Stk lvls:   4
97            **
98            **
99            ** Note: TRACEA can't destroy R3.  STORE passes the value stored
100           **         in the variable in R3 - this is needed by NEXT
101           **
102           ** Detail:
103           **    This routine is only called by the STORE routine, so the
104           **    only way to trace an assignment is to go through STORE.
105           **    The reason we need the address of the variable token in
106           **    the statement is that the destination variable name will
107           **    be reconstructed by decompiling the variable token.
108           **    If the destination is a substring or array element, the
109           **    subscript(s) will be computed from the information stored
110           **    in the STMTR0 & STMTR1.
111           **    If in a case the address of the variable token or the
112           **    content of STMTR0 & STMTR1 are lost, just zero the S-R1-2
113           **    to disable the tracing temporally.
114           **
115           ********************************************************************
116           ********************************************************************
117           *
118 0FA96 6763  DONtrc GOTO   TRCrtn
119           *
```

```
120             *
121 0FA9A 86D   =TRACEA ?ST=0  13              ARE WE RUNNING ?
122 0FA9D 9F            GOYES  DONtrc          DON'T TRACE IF NOT RUNNING
123 0FA9F 7694          GOSUB  D1=TRC          SEE WHICH TRACE MODE WE ARE IN
124 0FAA3 98A           ?A<=C  P               TRACE FLOW ONLY ?
125 0FAA6 0F            GOYES  DONtrc          RETURN IF SO
126             *
127             * Check if is in BASIC file
128             *
129 0FAA8 7793          GOSUB  BSTYCK
130 0FAAC 59E           GONC   DONtrc          Don't trace if not BASIC file
131             *
132 0FAAF 8F00          GOSBVL =fCALC?         Don't trace if in CALC mode
            000
133 0FAB6 4FD           GOC    DONtrc
134             *
135             * Check if we have a valid destination address ?
136             *
137 0FAB9 1B17          D0=(5) =S-R0-0
            8F2
138 0FAC0 146           C=DAT0 A
139 0FAC3 8AA           ?C=0   A
140 0FAC6 0D            GOYES  DONtrc          If not, don't trace
141             *-
142             *
143             ****************************************************************
144             * IN ORDER TO KNOW WHAT IS THE VARIABLE NAME OF THE DESTINATION,
145             * WE HAVE TO DECOMPILE THE LEFT-HAND SIDE OF THE ASSIGNMENT
146             * STATEMENT.
147 0FAC8 8F00          GOSBVL =D=AVME         D= AVMEME
            000
148 0FACF 8E00          GOSUBL =D0=AVS         A= AVMEMS
            00
149 0FAD5 DB            C=D    A               C= AVMEME
150 0FAD7 E2            C=C-A  A               C= AVAILABLE MEMORY IN NIBS
151 0FAD9 137           CD1EX                  SEE IF AT LEAST 16 NIBS AVAILABLE?
152 0FADC 1CF           D1=D1- 16
153 0FADF 46B           GOC    DONtrc           IF NOT, DON'T TRACE IT AT ALL
154             *
155 0FAE2 AF2           C=0    W               ZERO BEGINNING 8 BYTES OF BUFFER
156 0FAE5 1547          DAT0=C W
157 0FAE9 7C13          GOSUB  D1=SR1          GET SAVED ASSIGNMENT STMT ADDRESS
158 0FAED 8A8           ?A=0   A               WANT TO BE TRACED ?
159 0FAF0 6A            GOYES  DONtrc          No, if parse addr zero
160             *
161             * Check if this an UDF assignment such as :
162             *   10 FNX=45
163             *
164 0FAF2 848           ST=0   8
165 0FAF5 143           A=DAT1 A
166 0FAF8 8F00          GOSBVL =FNCK           Call function check routine
            000
167 0FAFF 5D1           GONC   TRC150          If carry clear, it is an "FNx="
168             *
169             * Check if this is a single line UDF such as 10 DEF FNF$="ABC".
```

```
170             * See if the statement is started with a tDEF ?
171             *
172 0FB02 136           CDOEX                   Save D0 in C
173 0FB05 8E00          GOSUBL =D0=PCA
          00
174 0FB0B 161           D0=D0+ 2
175 0FB0E 14A           A=DAT0 B
176 0FB11 136           CDOEX                   Restore D0 from C
177 0FB14 3100          LC(2)  =tDEF
178 0FB18 966           ?A#C   B                Is this a single line UDF ?
179 0FB1B D1            GOYES  TRC160           If not, decompile it by EXPRDC
180             *
181 0FB1D 3364  TRC150  LCASC  \NF\
          E4
182 0FB23 8F00          GOSBVL =OUT2TC          Output ASCII FN
          000
183 0FB2A 8F00          GOSBVL =VARDC+          Decompile the FN name
          000
184 0FB31 853           ST=1   3                Set Fn flag
185 0FB34 6D00          GOTO   TRC170
186             *
187 0FB38 8F00  TRC160  GOSBVL =EXPRDC          DECOMPILE EXPRESSION
          000
188 0FB3F 843           ST=0   3
189 0FB42 8E00  TRC170  GOSUBL =D0=RVS
          00
190 0FB48 1567          C=DAT0 W                SAVE 1ST 5 LETTERS OF THE LEFT-
191 0FB4C 109           R1=C                     HAND SIDE IN R1
192 0FB4F 863           ?ST=0  3                TRACING REGULAR VARIABLE ?
193 0FB52 90            GOYES  TRC180           IF SO, GOTO TRC180
194 0FB54 163           D0=D0+ 4                SKIP OVER THE "FN"
195 0FB57 1563          C=DAT0 X                READ THE FN NAME
196 0FB5B D5    TRC180  B=C    A
197 0FB5D 161           D0=D0+ 2
198 0FB60 14A           A=DAT0 B                SEE IF 2ND LETTER IS A DIGIT
199 0FB63 842           ST=0   2
200 0FB66 8F00          GOSBVL =DRANGE
          000
201 0FB6D 411           GOC    TRC210           IF NOT,GOTO TRC210
202 0FB70 852           ST=1   2                SET ALPHA DIGIT FLAG
203 0FB73 B25           B=B+1  XS               B(XS)= DIGIT+1
204 0FB76 161           D0=D0+ 2
205 0FB79 14A           A=DAT0 B
206 0FB7C 550           GONC   TRC215
207 0FB7F AA1   TRC210  B=0    XS
208 0FB82 840   TRC215  ST=0   0
209 0FB85 3142          LCASC  \$\
210 0FB89 966           ?A#C   B                IS THIS A STRING VARIABLE ?
211 0FB8C C0            GOYES  TRC220           IF NOT, GOTO TRC240
212 0FB8E 850           ST=1   0                SET STRING VARIABLE FLAG
213 0FB91 300           LCHEX  0               C(B) = 20
214 0FB94 0E69          B=B!C  B                OR IN THE STRING BIT TO NAME
215             *
216 0FB98 873   TRC220  ?ST=1  3                FUNCTION ?
217 0FB9B 03            GOYES  TRC245
```

```
218                *
219 0FB9D 7000             GOSUB  =ADRS40        LOOK FOR THE VARIABLE
220                *
221 0FBA1 844              ST=0   4              ASSUME IS NOT INDIRECT ADDRESS
222                *
223 0FBA4 310E   TRC230    LCHEX  E0
224 0FBA8 14A              A=DAT0 B
225 0FBAB 9E2              ?A<C   B              IS THIS AN INDIRECT ADDRESS ?
226 0FBAE A1               GOYES  TRC240         IF NOT, GOTO TRC240
227 0FBB0 31FF             LCHEX  FF
228 0FBB4 966              ?A#C   B
229 0FBB7 50               GOYES  TRC235
230                *
231 0FBB9 854              ST=1   4              THIS IS A INDIRECT SIMPLE STRING
232                *
233 0FBBC 161    TRC235   D0=D0+ 2
234 0FBBF 142              A=DAT0 A
235 0FBC2 130              D0=A
236 0FBC5 5ED              GONC   TRC230         (B.E.T.)
237 0FBC8 136    TRC240   CD0EX
238                *
239 0FBCB 108    TRC245   R0=C                  R0= VARIABLE ADDRESS
240 0FBCE 72F2             GOSUB  TRCLIN         WRITE "TRACE" & LINE # TO BUFFER
241 0FBD2 3102             LCASC  \ \
242 0FBD6 14C              DAT0=C B
243 0FBD9 161              D0=D0+ 2
244 0FBDC 7E82             GOSUB  TRFM20
245 0FBE0 7F23             GOSUB  DSBFC0         SET UP D0 AND D AGAIN
246 0FBE4 119              C=R1                  C= 1ST 5 LETTERS OF DESTINATION
247 0FBE7 15C9             DAT0=C 10             WRITE IT TO BUFFER
248 0FBEB 161              D0=D0+ 2              AT LEAST ONE LETTER OF DESTINATION
249 0FBEE 863              ?ST=0  3
250 0FBF1 50               GOYES  TRC246
251 0FBF3 163              D0=D0+ 4
252 0FBF6 862    TRC246   ?ST=0  2              STATIC VARIABLE ?
253 0FBF9 50               GOYES  TRC250         IF SO, GOTO TRC250
254 0FBFB 161              D0=D0+ 2              PASS THE "DIGIT"
255 0FBFE 860    TRC250   ?ST=0  0              NUMERIC VARIABLE ?
256 0FC01 50               GOYES  TRC260         IF SO, GOTO TRC260
257 0FC03 161              D0=D0+ 2              PASS THE "$" SIGN
258 0FC06 873    TRC260   ?ST=1  3              FUNCTION ?
259 0FC09 D2               GOYES  TRC265         IF SO, GOTO TRC345
260                *
261 0FC0B 874              ?ST=1  4              INDIRECT SIMPLE STRING ?
262 0FC0E 82               GOYES  TRC265
263                *
264 0FC10 118              C=R0
265 0FC13 135              D1=C                  D1 = VARIABLE ADDRESS
266 0FC16 147              C=DAT1 A
267 0FC19 D5               B=C    A
268 0FC1B A25              B=B+B  XS             Push out the stat bits
269 0FC1E A25              B=B+B  XS
270 0FC21 A25              B=B+B  XS             B(2) = Option base of the array
271                *
272 0FC24 31A0             LCHEX  0A
```

```
273 0FC28 985          ?B<C   P              A REAL ?
274 0FC2B B0           GOYES  TRC265         IF SO, DISPLAY "=" SIGN
275 0FC2D BE5          BSR    B
276 0FC30 A6D          B=B-1  B              SIMPLE VARIABLE ?
277 0FC33 560          GONC   TRC270         IF NOT, GOTO TRC270
278 0FC36 61D0  TRC265 GOTO   TRC345
279
280            *
281            * ARRAY OR VECTOR
282            * THE SUBSCRIPT(S) IS COMPUTED FROM THE ARRAY REGISTER NUMBER
283            * SAVED IN S-R1-0. THE REGISTER # IS COUNTED FROM ZERO.
284            * THE FORMULAR TO COMPUTE THE SUBSCRIPT IS :
285            * 1. ONE DIMENSION ARRAY :
286            *      OPTION BASE 0:  i = REGISTER #
287            *           //     1:  i = REGISTER + 1
288            * 2. TWO DIMENSIONS ARRAY:  ( C IS THE 2ND SUBSCRIPT LIMIT)
289            *      OPTION BASE 0:  i = REGISTER # / (C+1)
290            *                      j = REMAINDER OF ABOVE DIVISION
291            *           //     1:  i = (REGISTER # / C) + 1
292            *                      j = REMAINDER + 1
293            *
294 0FC3A 3182  TRC270 LCASC  \(\
295 0FC3E 14C          DAT0=C B
296 0FC41 161          D0=D0+ 2              PASS THE "("
297 0FC44 133          AD1EX
298 0FC47 1F18         D1=(5) =S-R1-0        POINTS TO SAVED REGISTER #
          8F2
299 0FC4E AF2          C=0    W
300 0FC51 147          C=DAT1 A
301 0FC54 96D          ?B#0   B              TWO DIMENSION ARRAY ?
302 0FC57 26           GOYES  TRC290         IF SO, GOTO TRC290
303 0FC59 DA           A=C    A              A= REGISTER #
304 0FC5B 929          ?B=0   XS             OPTION BASE 0 ?
305 0FC5E 40           GOYES  TRC275         IF SO, GOTO TRC340
306 0FC60 E4           A=A+1  A
307            * FOR SUB-STRING ASSIGNMENT, WE WILL LOSE THE REGISTER NUMBER IF
308            * ONE OF THE SUBSCRIPT IS AN USER-DEFINED FUNCTION, LIKE
309            * A$(3)[12,FNX] = "ABC".
310            *
311 0FC62 3400  TRC275 LCHEX  10000
          001
312 0FC69 8B2          ?A<C   A
313 0FC6C 94           GOYES  TRC285
314 0FC6E 110          A=R0
315 0FC71 131          D1=A                  D1 @ VARIABLE DOPE VECTOR
316 0FC74 17A          D1=D1+ 11
317 0FC77 143          A=DAT1 A              A = ARRAY POINTER
318 0FC7A 137          CD1EX
319 0FC7D E2           C=C-A  A              C = ARRAY START ADDRESS
320 0FC7F 1F17         D1=(5) =S-R0-0
          8F2
321 0FC86 143          A=DAT1 A              A = ASSIGNMENT DESTINATION
322 0FC89 100          R0=A
323 0FC8C 17F          D1=D1+ 16
324 0FC8F 174          D1=D1+ 5              D1 @ S-R1-1 (CONTAIN MAX.STR.LEN)
```

```
325 OFC92 92D          ?B#0   XS
326 OFC95 20           GOYES  TRC276
327 OFC97 D1    TRC276 B=0    A
328 OFC99 440          GOC    TRC280          IF OPTION BASE =1, START FROM 1
329 OFC9C CD           B=B-1  A               IF OPTION BASE =0, START FROM 0
330 OFC9E E5    TRC280 B=B+1  A
331 OFCA0 143          A=DAT1 A               A = MAX. STRING LENGTH IN BYTES
332 OFCA3 E4           A=A+1  A
333 OFCA5 E4           A=A+1  A
334 OFCA7 C4           A=A+A  A
335 OFCA9 C2           C=C+A  A               C = ADDRESS OF NEXT STRING ELEMENT
336 OFCAB 110          A=R0
337 OFCAE 8BE          ?A>=C  A
338 OFCB1 DE           GOYES  TRC280
339 OFCB3 D4           A=B    A
340 OFCB5 6440  TRC285 GOTO   TRC340          PUT ")" AND "=" TO BUFFER (B.E.T.)
341             * TWO DIMENSION ARRAY
342 OFCB9 131   TRC290 D1=A
343 OFCBC 172          D1=D1+ 3
344 OFCBF AF0          A=0    W
345 OFCC2 15B3         A=DAT1 4               A= 2ND SUBSCRIPT LIMIT
346 OFCC6 DE           ACEX   A               A= REGISTER #, C= 2ND SUB.LIMIT
347 OFCC8 92D          ?B#0   XS              OPTION BASE 1 ?
348 OFCCB D0           GOYES  TRC310          IF SO, GOTO TRC310
349 OFCCD E6           C=C+1  A               OPTION BASE 0
350 OFCCF 8E00         GOSUBL =IDIV           DIVIDE A BY C
            00
351 OFCD5 5C0          GONC   TRC320          A= 1ST SUBSCRIPT, B= 2ND SUBSCRIPT
352 OFCD8 8E00  TRC310 GOSUBL =IDIV
            00
353 OFCDE E4           A=A+1  A               A= 1ST SUBSCRIPT
354 OFCE0 E5           B=B+1  A               B= 2ND SUBSCRIPT
355 OFCE2 20    TRC320 P=     0
356 OFCE4 D9           C=B    A
357 OFCE6 108          R0=C
358 OFCE9 7362         GOSUB  DSHEX           WRITE 1ST SUBSCRIPT TO BUFFER
359 OFCED 31C2         LCASC  \,\             WRITE COMMA TO BUFFER
360 OFCF1 14C          DAT0=C B
361 OFCF4 161          D0=D0+ 2
362 OFCF7 110          A=R0
363 OFCFA 7252  TRC340 GOSUB  DSHEX           WRITE 2ND SUBSCRIPT TO BUFFER
364 OFCFE 3192         LCASC  \)\             WRITE ")" TO BUFFER
365 OFD02 14C          DAT0=C B
366 OFD05 161          D0=D0+ 2
367 OFD08 870   TRC345 ?ST=1  0               IS A STRING VARIABLE ?
368 OFD0B 91           GOYES  TRC400          IF SO, GOTO TRC400
369 OFD0D 31D3  TRC350 LCASC  \=\             WRITE "=" TO BUFFER
370 OFD11 14C          DAT0=C B
371 OFD14 161          D0=D0+ 2
372 OFD17 7351  TRC360 GOSUB  TRFM20          WRITE END OF BUFFER MARK TO BUFFER
373 OFD1B 860          ?ST=0  0               NUMERIC VARIABLE ?
374 OFD1E F5           GOYES  TRC450          IF SO, GOTO TRC450, DISPLAY RESULT
375 OFD20 69D0  TRC370 GOTO   TRC490          GOTO SEND CR/LF TO DISPLAY
376             *
377             * FOR STRING VARIABLE TO DISPLAY THE SUBSTRING SUBSCRIPT
```

```
378                *
379 0FD24 1F67     TRC400 D1=(5) =S-R0-1        POINTS TO 1ST SUBSCRIPT
          8F2
380 0FD2B AF0             A=0    W
381 0FD2E 143             A=DAT1 A              A = 1ST STRING SUBSCRIPT
382 0FD31 174             D1=D1+ 5
383 0FD34 AF2             C=0    W
384 0FD37 147             C=DAT1 A              C = 2ND STRING SUBSCRIPT
385 0FD3A 8AC             ?A#0   A              1ST SUBSCRIPT = 0 ?
386 0FD3D B0              GOYES  TRC410         IF NOT, GOTO TRC410
387 0FD3F D5              B=C    A              IF 1ST SUB=0 & 2ND SUB=FFFFE
388 0FD41 E5              B=B+1  A               IT MEANS ENTIRE STRING
389 0FD43 E5              B=B+1  A
390 0FD45 41D             GOC    TRC360         ENTIRE STRING ?
391 0FD48 108    TRC410   R0=C                  R0 = 2ND SUBSCRIPT
392 0FD4B 31B5            LCASC  \[\
393 0FD4F 81C             ASRB
394 0FD52 E4              A=A+1  A
395 0FD54 72F1            GOSUB  DSHEX+         WRITE 1ST SUBSCRIPT TO BUFFER
396 0FD58 110             A=R0
397 0FD5B D6              C=A    A
398 0FD5D E6              C=C+1  A              2ND SUBSCRIPT = FFFFE ?
399 0FD5F E6              C=C+1  A
400 0FD61 4D0             GOC    TRC420         IF SO, DON'T DISPLAY 2ND SUBSCRIPT
401 0FD64 31C2            LCASC  \,\            WRITE COMMA TO BUFFER
402 0FD68 81C             ASRB
403 0FD6B 78D1            GOSUB  DSHEX+
404 0FD6F 31D5   TRC420   LCASC  \]\            WRITE "]" TO BUFFER
405 0FD73 14C             DAT0=C B
406 0FD76 161             D0=D0+ 2
407 0FD79 6D9F            GOTO   TRC360
408                *
409                * DISPLAY THE RESULT, RECALL IT FROM WHERE IT IS STORED.
410                *
411 0FD7D 1B17   TRC450 D0=(5) =S-R0-0
          8F2
412 0FD84 1527            A=DAT0 W
413 0FD88 130             D0=A                  D0 @ DESTINATION
414 0FD8B 8E00            GOSUBL =D1mstk
          00
415 0FD91 810             ASLC   W              A(0)=VAR TYPE(2,1,0,D,E-I,S,R,CMS,
416 0FD94 30C             LCHEX  C                           CM)
417 0FD97 B02             C=C-A  P              C(0)=A,B,C,F,E - I,S,R,CMS,CM)
418 0FD9A DA              A=C    A
419 0FD9C 30F             LCHEX  F
420 0FD9F B02             C=C-A  P              C(0)=5,4,3,2,1 - I,S,R,CMS,CM
421 0FDA2 8E00            GOSUBL =DATLEN        C(A)=HEX 6,9,10,12,20- I,S,R,CMS,CM
          00
422 0FDA8 845             ST=0   5              Clear error flag
423 0FDAB 843             ST=0   3              Clear complex flag
424 0FDAE 8E00            GOSUBL =RCLNUM        RECALL THE NUMBER
          00
425 0FDB4 875             ?ST=1  5
426 0FDB7 34              GOYES  TRC490         IF NO ROOM, DON'T DISPLAY NUMBER
427 0FDB9 863             ?ST=0  3              COMPLEX NUMBER ?
```

```
428 OFDBC C0          GOYES  TRC480       IF NOT, GOTO TRC480
429 OFDBE 31E0        LCHEX  0E
430 OFDC2 1C1         D1=D1- 2
431 OFDC5 14D         DAT1=C B
432             *
433 OFDC8 8F00  TRC480 GOSBVL =STR$SB
          000
434 OFDCF 8E00        GOSUBL =ave=d1
          00
435             *
436             *
437             * NOW D1 POINTS TO NEW TOP OF STACK WHICH SHOULD BE A STRING
438             * AND SINCE THE STRING IS IN REVERSE ORDER, SO WE HAVE TO SEND
439             * THE STRING STARTING FROM THE END.
440             *
441 OFDD5 AF0           A=0    W
442 OFDD8 8E00        GOSUBL =REVPOP        REVERSE THE STRING ON STACK
          00
443 OFDDE 81C         ASRB
444 OFDE1 8E00        GOSUBL =DSPCNA        SEND IT TO DISPLAY
          00
445 OFDE7 8E00        GOSUBL =D1=stk
          00
446 OFDED 8F00        GOSBVL =POPMTH
          000
447 OFDF4 8E00        GOSUBL =ave=d1
          00
448             *
449 OFDFA 7400  TRC490 GOSUB  crlfsd        Send CR/LF
450 OFDFE 6000  TRCrtn GOTO   =TRCRTN
451             *
452             * SEND "CR" & "LF" AT THE END
453             *
454 OFE02 8D00 =crlfsd GOVLNG =CRLFSD
          000
455             *
456 OFE09 1FB8  D1=SR1 D1=(5) =S-R1-2
          8F2
457 OFE10 143         A=DAT1 A
458 OFE13 131         D1=A
459 OFE16 01          RTN
460             *
461             *
462             ********************************************************************
463             ********************************************************************
464             **
465             ** Name:    TRFLCK  -  Trace mode check
466             ** Name:    TRFCK-  -  Trace mode check
467             **
468             ** Category:   EXCUTL
469             **
470             ** Purpose:  To check if need to trace flow
471             **
472             ** Entry :
473             **    P=0.
```

```
                **    TRFLCK:  Not executing RESTORE
                **    TRFCK=:  S10 = 1 Must be executing RESTORE (no trace)
                **                   = 0 Not executing RESTORE
                ** Exit:
                **    Carry set => No trace
                **    Carry clear => Trace needed
                **    TRFLCK:  S10 = 0
                **
                **
                ** Uses:  A(A),C(A),D1, S10
                **
                ** Calls: D1=TRC, BSTYCK
                **
                ** Stk lvls:  2
                **
                ** Detail:
                **    Only need to trace flow if all the following conditions
                **    are met:
                **    1. In trace flow mode (S15=1 and TRACEM=2 or 6)
                **    2. Executing program (S13=1).
                **    3. Not executing RESTORE (S10=0).
                **
                ********************************************************************
                ********************************************************************
                *
OFE18 84A       =TRFLCK ST=0   10            DON'T CARE ABOUT RESTORE
OFE1B 86F       =TRFCK- ?ST=0  15            TRACING ?
OFE1E 00                RTNYES               IF NOT, DON'T TRACE
OFE20 86D               ?ST=0  13            RUNNING ?
OFE23 00                RTNYES
OFE25 87A               ?ST=1  10            FOR RESTORE COMMAND ?
OFE28 00                RTNYES               RETURN IF SO
OFE2A 7B01              GOSUB  D1=TRC        SEE WHICH TRACE MODE WE ARE IN
OFE2E 0E06              A=A&C  P
OFE32 A0C               A=A-1  P             SET CARRY IF TRACE OFF
OFE35 400               RTNC
                *
                * See if type BASIC
                *
OFE38 7700              GOSUB  BSTYCK
OFE3C 440               GOC    CLCRY
OFE3F 02                RTNSC
OFE41 03        CLCRY   RTNCC
                **
                ********************************************************************
                ********************************************************************
                **
                ** Name:    BSTYCK  -  Check if Current File is BASIC
                **
                ** Category:   FILUTL
                **
                ** Purpose: Check if the file type is BASIC
                **
                ** Entry: CURRST points to current file start
                **        P=0
```

```
                **
                ** Exit: Carry set => Current file is type BASIC
                **       Carry clear => Current file is not type BASIC
                **
                ** Uses: A(A),C(A), D1
                **
                ** Calls: D1=CRS
                **
                ** Stk lvls: +1
                **
                ** Detail:
                **       Find the current file header by CURRST, then check
                **       the file type in the file header.
                ********************************************************************
                ********************************************************************
                *
0FE43 8F00 =BSTYCK GOSBVL =D1=CRS
      000
0FE4A 17E          D1=D1+ (oFTYPh)-1
0FE4D 143          A=DAT1 A
0FE50 F4           ASR    A
0FE52 8D00         GOVLNG =BASTYP
      000
                *
                *
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) TRFROM  -  Trace Line Number
                **
                ** Category:   EXCUTL
                **
                ** Purpose:  Routine to generate the "Trace nnnn to" in display.
                **           The current line number is computed from PCADDR.
                **
                ** Entry:  PCADDR @ current line length
                **         P=0
                **
                ** Exit:   Send "Trace nnnn to" to display buffer
                **         (Via AVS2DS)
                **
                ** Calls:  TRCLIN
                **
                ** Uses:  A,B,C,D,D0,D1,D0, R0, P
                **
                ** Stk lvls:  +4
                **
                ** Note:  Will exit to error routine if not enough memory to
                **        buffer the display line.
                **
                ********************************************************************
                ********************************************************************
                *
0FE59 7760 =TRFROM GOSUB  TRCLIN
0FE5D 3702         LCASC  \ ot \
```

```
          47F6
          02
   582 0FE67 15C7         DATO=C 8
   583 0FE6B 167          DO=DO+ 8
   584 0FE6E 31FF =TRFM20 LCHEX  FF
   585 0FE72 14C          DATO=C B
   586 0FE75 8C00         GOLONG =AVS2DS
          00
   587                 *
   588                 ******************************************************************
   589                 ******************************************************************
   590                 **
   591                 ** Name:    TRTO    -  Generate Trace Message
   592                 ** Name:(S) TRTO+   -  Generate Trace Message
   593                 ** Name:    TRTO-   -  Generate Trace Message
   594                 ** Name:    TRTO*   -  Generate Trace Message
   595                 **
   596                 ** Category:   EXCUTL
   597                 **
   598                 ** Purpose:  Generates "to nnnn" for TRACE FLOW mode.
   599                 **           The line number is computed from DO on entry.
   600                 **
   601                 ** Entry :
   602                 **        DO is pointing at some where in the current line.
   603                 **        (A line can have multiple statements)
   604                 **
   605                 **    TRTO+:  DO pts at EOL/@ preceding a statement
   606                 **            P=0
   607                 **    TRTOr:  DO pts at the line length of a statement.
   608                 **    TRTO-:  DO pts at middle of a statement
   609                 **    TRTO*:  DO pts at EOL preceding the current line.
   610                 **
   611                 ** Exit:  Via CRLFSD
   612                 **    TRTO+: R1 = DO on entry.
   613                 **
   614                 ** Calls:  CPL#10, DO=PCA, DSBFCK, DSINTR, TRFM20.
   615                 **
   616                 ** Uses:
   617                 **    A,B,C,D,DO,D1,S9
   618                 **    TRTO+ also uses R1 to save the DO on entry.
   619                 **
   620                 ** Stk lvls:  +5
   621                 **
   622                 ** History:
   623                 **    Date    Programmer   Modification
   624                 **   -------  ----------   --------------------------------
   625                 **   3/11/83   SC          Document
   626                 **
   627                 ******************************************************************
   628                 ******************************************************************
   629                 **
   630 0FE7B 136  =TRTO+  CDOEX                 DO AT EOL/@ PRECEEDING THE LINE
   631 0FE7E 109          R1=C
   632 0FE81 134          DO=C
   633 0FE84 14A          A=DATO B
```

```
634 0FE87 908          ?A=0   P              POINTS AT AN EOL ?
635 0FE8A F1           GOYES  TRTO*
636 0FE8C 161  =TRTOr  D0=D0+ 2
637 0FE8F 8F00 =TRTO-  GOSBVL =CPL#10
          000
638 0FE96 521          GONC   TRTO*          CARRY CLEAR IF LINE # FOUND
639               *
640 0FE99 8E00         GOSUBL =D0=PCA        TRY PCADDR
          00
641 0FE9F 8F00         GOSBVL =CPL#10
          000
642 0FEA6 491          GOC    EOFIL          IF STILL NO LINE #, GIVE UP
643               *
644 0FEA9 161  =TRTO*  D0=D0+ 2              POINTS TO LINE #
645 0FEAC 15A3  TRTO10 A=DAT0 4
646 0FEB0 D8           B=A    A
647 0FEB2 7D50         GOSUB  DSBFCO
648 0FEB6 D4           A=B    A
649 0FEB8 7C90         GOSUB  DSINTR
650 0FEBC 7EAF =TRTOEN GOSUB  TRFM20
651 0FEC0 614F  EOFIL  GOTO   crlfsd
652               *
653               ********************************************************************
654               ********************************************************************
655               **
656               ** Name:    TRCLIN  -  Display "Trace line nnnn" Message
657               **
658               ** Category:   EXCUTL
659               **
660               ** Purpose:  Generates "Trace line nnnn" to display.
661               **           The current line number is computed from PCADDR.
662               **
663               ** Entry:  PCADDR pts at current statement length
664               **         P=0
665               **
666               ** Exit:   Carry clear
667               **         Exit to error routine if not enough memory to
668               **         buffer the display line.
669               **
670               ** Calls:  COMPL#, LIN#DC, DSBFCK, D0=PCA
671               **
672               ** Uses:  A,B,C,D,D0,D1,S9,P
673               **
674               ** Stk lvls: +3
675               **
676               ** History:
677               **
678               **  Date     Programmer    Modification
679               ** -------   ----------    ----------------------------------
680               ** 3/11/83   SC            Document
681               ********************************************************************
682               ********************************************************************
683               *
684 0FEC4 D2   =TRCLIN C=0    A
685 0FEC6 20           P=     0
```

```
686 0FEC8 31A3          LC(2)  58
687 0FECC 7540          GOSUB  DSBFCK          SEE IF ROOM FOR DISPLAY BUFFER
688 0FED0 3F45  TRFM10  LCASC  \il ecarT\
          2716
          3656
          02C6
          96
689 0FEE2 1547          DAT0=C W               WRITE "TRACE LI" TO BUFFER
690 0FEE6 16F           DO=DO+ 16
691 0FEE9 35E6          LCASC  \ en\
          5602
692 0FEF1 15C5          DAT0=C 6
693 0FEF5 8E00          GOSUBL =DO=PCA
          00
694 0FEFB 8F00          GOSBVL =COMPL#         COMPUTE CURRENT LINE #
          000
695             *
696 0FF02 7D00          GOSUB  DSBFCO
697 0FF06 16A           DO=DO+ 11
698 0FF09 16A           DO=DO+ 11              POINTS TO END OF BUFFER
699 0FF0C 8D00          GOVLNG =LIN#DC         PUT CURRENT LINE # TO BUFFER
          000
700             *
701             ********************************************************************
702             ********************************************************************
703             **
704             ** Name:    DSBFCK  -  Check Free Space on Math Stack
705             **
706             ** Category:   EXCUTL
707             **
708             ** Purpose:  Check if there is enough free space on the math
709             **           stack can be used for display buffer.
710             **
711             ** Entry:  C(A) = Number of nibbles needed
712             **
713             ** Exit:   Carry set
714             **         DO @ Available memory start (AVMEMS).
715             **         D(A) = FORSTK
716             **         A(A) = Address of available memory start.
717             **
718             **         Exit to NOMEM error routine if not enough memory to
719             **         buffer the display line.
720             **
721             ** Uses:  A(A), C(A), D(A), DO
722             **
723             ** Stk lvls: 0
724             **
725             ** History:
726             **
727             **  Date     Programmer    Modification
728             ** -------   ----------    ----------------------------------
729             ** 3/11/83   SC            Document
730             ********************************************************************
731             ********************************************************************
732             *
```

```
733 OFF13 D2    DSBFCO C=0    A
734 OFF15 DE    DSBFCK ACEX   A
735 OFF17 1BE9         DO=(5) =FORSTK        NEED BUFFER AT LEAST 32 NIBS ON
          5F2
736 OFF1E 146          C=DAT0 A               OPERATING STACK
737 OFF21 D7           D=C    A
738 OFF23 E2           C=C-A  A
739 OFF25 189          DO=DO- (TFORN)-(AVMEMS)
740 OFF28 142          A=DAT0 A
741 OFF2B 130          DO=A
742 OFF2E 8B2          ?C>A   A
743 OFF31 00           RTNYES
744 OFF33 8C00 =MEMERk GOLONG =NOMEM
          00
745               *
746               *
747 OFF39 1F0B  D1=TRC D1=(5) =TRACEM
          7F2
748 OFF40 20           P=     0
749 OFF42 302          LCHEX  2
750 OFF45 14B          A=DAT1 B
751 OFF48 01           RTN                   PRESERVE CARRY, DON'T USE RTNCC
752               *
753               ********************************************************************
754               ********************************************************************
755               **
756               ** Name:    DSHEX   -  Convert a Hex to Dec and Displat it.
757               **
758               ** Category:   EXCUTL
759               **
760               ** Purpose:  Convert an Hex number to decimal number and display
761               **           it.
762               **
763               ** Entry:  A(A) = The hex number
764               **         DO @ Next character position in display buffer.
765               **
766               ** Exit:   Via LIN#AU
767               **
768               ** Uses:  A,B,C,D(S),P
769               **
770               ** Stk lvls: +2
771               **
772               ** History:
773               **
774               **  Date     Programmer    Modification
775               ** -------   -----------   ----------------------------------
776               ** 3/11/83   SC            Document
777               ********************************************************************
778               ********************************************************************
779               *
780 OFF4A 14C   DSHEX+ DAT0=C B
781 OFF4D 161          DO=DO+ 2
782 OFF50 8E00  DSHEX  GOSUBL =HEXDEC
          00
783 OFF56 04           SETHEX
```

```
    784 OFF58 8D00  DSINTR GOVLNG =LIN#AU
              000
    785             *
    786             *
    787 OFF5F               END
```

ADRS40	Ext			-	219							
AVMEMS	Abs	193940	#2F594	-	12	739						
AVS2DS	Ext			-	586							
BASTYP	Ext			-	549							
=BSTYCK	Abs	65091	#0FE43	-	545	129	513					
CLCRY	Abs	65089	#0FE41	-	516	514						
COMPL#	Ext			-	694							
CPL#10	Ext			-	637	641						
CRLFSD	Ext			-	454							
D0=AVS	Ext			-	148	189						
D0=PCA	Ext			-	173	640	693					
D1=CRS	Ext			-	545							
D1=SR1	Abs	65033	#0FE09	-	456	157						
D1=TRC	Abs	65337	#0FF39	-	747	55	123	506				
D1mstk	Ext			-	414	445						
D=AVME	Ext			-	147							
DATLEN	Ext			-	421							
DONtrc	Abs	64150	#0FA96	-	118	122	125	130	133	140	153	159
DRANGE	Ext			-	200							
DSBFCO	Abs	65299	#0FF13	-	733	245	647	696				
DSBFCK	Abs	65301	#0FF15	-	734	687						
DSHEX	Abs	65360	#0FF50	-	782	358	363					
DSHEX+	Abs	65354	#0FF4A	-	780	395	403					
DSINTR	Abs	65368	#0FF58	-	784	649						
DSPCNA	Ext			-	444							
EOFIL	Abs	65216	#0FEC0	-	651	642						
EXPRDC	Ext			-	187							
FNCK	Ext			-	166							
FORSTK	Abs	193950	#2F59E	-	12	735						
HEXDEC	Ext			-	782							
IDIV	Ext			-	350	352						
LIN#AU	Ext			-	784							
LIN#DC	Ext			-	699							
=MEMERk	Abs	65331	#0FF33	-	744							
NOMEM	Ext			-	744							
NXTSTM	Ext			-	64							
OUT2TC	Ext			-	182							
POPMTH	Ext			-	446							
RCLNUM	Ext			-	424							
REVPOP	Ext			-	442							
S-R0-0	Abs	194673	#2F871	-	12	137	320	411				
S-R0-1	Abs	194678	#2F876	-	12	379						
S-R1-0	Abs	194689	#2F881	-	12	298						
S-R1-2	Abs	194699	#2F88B	-	12	456						
STR$SB	Ext			-	433							
SelTrc	Abs	1	#00001	-	40							
TFORN	Abs	193950	#2F59E	-	12	739						
TRACDC	Ext			-	43							
=TRACE	Abs	64081	#0FA51	-	45							
=TRACEA	Abs	64154	#0FA9A	-	121							
TRACEM	Abs	194480	#2F7B0	-	12	747						
TRACEP	Ext			-	44							
TRAL20	Abs	64140	#0FA8C	-	63	61						
TRC150	Abs	64285	#0FB1D	-	181	167						
TRC160	Abs	64312	#0FB38	-	187	179						

```
 TRC170  Abs   64322 #0FB42 -   189   185
 TRC180  Abs   64347 #0FB5B -   196   193
 TRC210  Abs   64383 #0FB7F -   207   201
 TRC215  Abs   64386 #0FB82 -   208   206
 TRC220  Abs   64408 #0FB98 -   216   211
 TRC230  Abs   64420 #0FBA4 -   223   236
 TRC235  Abs   64444 #0FBBC -   233   229
 TRC240  Abs   64456 #0FBC8 -   237   226
 TRC245  Abs   64459 #0FBCB -   239   217
 TRC246  Abs   64502 #0FBF6 -   252   250
 TRC250  Abs   64510 #0FBFE -   255   253
 TRC260  Abs   64518 #0FC06 -   258   256
 TRC265  Abs   64566 #0FC36 -   278   259   262   274
 TRC270  Abs   64570 #0FC3A -   294   277
 TRC275  Abs   64610 #0FC62 -   311   305
 TRC276  Abs   64663 #0FC97 -   327   326
 TRC280  Abs   64670 #0FC9E -   330   328   338
 TRC285  Abs   64693 #0FCB5 -   340   313
 TRC290  Abs   64697 #0FCB9 -   342   302
 TRC310  Abs   64728 #0FCD8 -   352   348
 TRC320  Abs   64738 #0FCE2 -   355   351
 TRC340  Abs   64762 #0FCFA -   363   340
 TRC345  Abs   64776 #0FD08 -   367   278
 TRC350  Abs   64781 #0FD0D -   369
 TRC360  Abs   64791 #0FD17 -   372   390   407
 TRC370  Abs   64800 #0FD20 -   375
 TRC400  Abs   64804 #0FD24 -   379   368
 TRC410  Abs   64840 #0FD48 -   391   386
 TRC420  Abs   64879 #0FD6F -   404   400
 TRC450  Abs   64893 #0FD7D -   411   374
 TRC480  Abs   64968 #0FDC8 -   433   428
 TRC490  Abs   65018 #0FDFA -   449   375   426
=TRCLIN  Abs   65220 #0FEC4 -   684   240   580
 TRCRTN  Ext                 -   450
 TRCrtn  Abs   65022 #0FDFE -   450   118
=TRFCK-  Abs   65051 #0FE1B -   500
=TRFLCK  Abs   65048 #0FE18 -   499
 TRFLOW  Abs   64129 #0FA81 -    58    56
 TRFM10  Abs   65232 #0FED0 -   688
=TRFM20  Abs   65134 #0FE6E -   584   244   372   650
=TRFROM  Abs   65113 #0FE59 -   580
 TROFF   Abs   64120 #0FA78 -    55    53
=TRTO*   Abs   65193 #0FEA9 -   644   635   638
=TRTO+   Abs   65147 #0FE7B -   630
=TRTO-   Abs   65167 #0FE8F -   637
 TRTO10  Abs   65196 #0FEAC -   645
=TRTOEN  Abs   65212 #0FEBC -   650
=TRTOr   Abs   65164 #0FE8C -   636
 TRVARS  Abs   64117 #0FA75 -    54    49
 VARDC+  Ext                 -   183
 ave=d1  Ext                 -   434   447
=crlfsd  Abs   65026 #0FE02 -   454   449   651
 fCALC?  Ext                 -   132
 oFTYPh  Abs      16 #00010 -    13   546
 tDEF    Ext                 -   177
```

```
 tFLOW   Abs 2687471 #901EF -    13    51
 tVARS   Abs 5964271 #B01EF -    13    46
```

Input Parameters

  Source file name is SC&TRC::MS

  Listing file name is SC/TRC:TI:ML::-1

  Object file name is SC%TRC:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *       J  PPPP    &     M   M  EEEEE  M   M
 2            *       J  P   P  & &    MM MM  E      MM MM
 3            *       J  P   P  & &    M M M  E      M M M
 4            *       J  PPPP    &     M M M  EEEE   M M M
 5            *       J  P      & & &  M   M  E      M   M
 6            *    J  J  P      &  &   M   M  E      M   M
 7            *     JJJ   P      && &  M   M  EEEEE  M   M
 8            *
 9                      TITLE  Program Edit <831212.1204>
10 OFF5F               ABS    #OFF5F
11                     RDSYMB SB%RAM::MS
12                     RDSYMB TI%EQU::MS
13           ****************************************************************
14           ****************************************************************
15           **
16           ** Name:(S) PEDIT   -  Program Edit
17           ** Name:(S) PEDITD  -  Program Edit to delete line
18           ** Name:    PEDITM  -  Program Edit not collapsing stacks
19           **
20           ** Category:   FILUTL
21           **
22           ** Purpose:    Edit/delete line in current program
23           **
24           ** Entry:      PEDIT  =>
25           **                     Edit line into current program
26           **                     Line in output buffer
27           **                      S8 is cleared
28           **                     Stacks/SUSP prog cleared after
29           **                       Protection Check
30           **                 PEDITD =>
31           **                     S8 must be set
32           **                     Delete Line
33           **                     Line# to delete in output buffer
34           **                     Stacks/SUSP prog cleared after
35           **                       Protection Check
36           **                 PEDITM =>
37           **                     MERGE command entry point
38           **                     S8 must be clear, to avoid delete
39           **                     PRIVATE and SECURE have already been
40           **                       checked
41           **                     Stacks will NOT be collapsed
42           **
43           ** Exit:
44           **           Carry Clear
45           **              R3= offset of memory at higher address
46           **              Memory pointers updated
47           **           else
48           **              Error Exit
49           **               Non BASIC file type          eFTYPE
50           **               File protected               ePROT
51           **               Unsuccessful replace of line
52           **
53           ** Calls:    FINDL+, SAVEL+, RPLLIN, OBCOLL, CHKPSF, CLPSTK
54           **           NXTLIN, D1=CRS, DOOUTB, CLLINK
55           **
```

```
                ** Uses:     A, B, C, D, D0, D1, OUTBS, R0-R3, S8
                **           If GOTO/GOSUB links are cleared, S1 is used
                **
                ** Detail:
                **
                **  PEDIT:   Clear Delete Line flag
                **  PEDITD:  If current file type not BASIC or protected
                **              Error Exit
                **           Colapse stack, zero addresses, clear SUSP annun.
                **  PEDITM:  Zero Label chain and all GOTO links in file
                **              Move Output Buffer to end of available memory
                **           Set D0 @ start of line to Edit (@ OUTBS)
                **           Update CURRL to new line #  (SAVELO)
                **           If null line  (S8=1)
                **              Collapse Output Buffer
                **           Call FINDL to find a match on line# >=
                **           Set D = End of program memory  (MAINEN)
                **           Compute old line length
                **           Replace line
                **           If unsuccessful
                **             THEN MFERR
                **
                ** Stack lvls: 5
                **
                ** History:
                **
                **    Date     Programmer    Modifications
                **  --------  ----------    -------------
                **  07/08/82  SW            Updated documentation
                **  01/11/83  SW            Eliminated poll on non-RAM device
                **  03/02/83  JP            Packed GETPRe to CHKPSF
                **  03/03/83  JP            Moved PEDITM entry, CLPSTK call
                **
                ***************************************************************
                ***************************************************************
                *
                *
OFF5F 848       =PEDIT  ST=0   8              Clear NULL line flag
                *
                * Delete Line Entry
                *   Assumes S8 set to indicate "delete" line
                *
                * Check File type, PRIVATE, SECURE
                *   Error exit from CHKPSF
                * Collapse stacks, clear program information
                *
OFF62           =PEDITD
OFF62 8F00       PEDT00 GOSBVL =CHKPSF
      000
OFF69 8F00              GOSBVL =CLPSTK
      000
                *
                * MERGE entry point:
                *   Avoids collapsing of stacks/ clearing program memory
                *   Assumes MERGE has check protection
```

```
109                *
110                * Zero Label Chain head; all GOTO/GOSUB links
111                *   D1 @ File Type field of file header
112                *
113 0FF70          =PEDITM
114 0FF70 8F00      PEDIT0 GOSBVL =CLLINK
           000
115                *
116                * Set D0,D1 @ start of Output Buffer (@ line#) to PEDIT
117                * Update CURRL, Save line#
118                * If Null line
119                *    Collapse the Output Buffer
120                * Try to find new line# within current file
121                *
122 0FF77 8F00             GOSBVL =D0OUTB        Set D0 = OUTBS
           000
123 0FF7E 73F1             GOSUB  SAVEL+         Read line#, update CURRL
124 0FF82 D5               B=C    A              Save line#
125 0FF84 868              ?ST=0  8              Not Null line ?
126 0FF87 90               GOYES  PEDIT2
127 0FF89 8F00             GOSBVL =OBCOLL        Collapse Output Buffer
           000
128 0FF90 7250   PEDIT2    GOSUB  FINDL+         Find line# in B(A)
129                *
130                * Set Old line length = 0
131                * If line# found
132                *    Save start of line
133                *    Skip to End of line
134                *    Compute Total line length
135                *
136 0FF94 D1               B=0    A              Set Old Line Length = 0
137 0FF96 501              GONC   PEDIT3         Line # not found
138 0FF99 133              AD1EX                 A <-- start of line
139 0FF9C D8               B=A    A              B <-- start of line
140 0FF9E 133              AD1EX
141 0FFA1 7C80             GOSUB  NXTLIN         C <-- end of line
142 0FFA5 ED               B=C-B  A              Total line length
143                *
144                * Entry Conditions to RPLLIN:
145                *   R3   = Old line length
146                *        = 0 if old line not found
147                *   C(A) = File start
148                *   A(A) = End of old line   (place to insert)
149                *
150 0FFA7 D9     PEDIT3    C=B    A
151 0FFA9 10B              R3=C                  R3 <-- Old line length
152 0FFAC 133              AD1EX                 A  <-- End of old line
153 0FFAF 8F00             GOSBVL =D1=CRS        C(A) <-- CURRST
           000
154 0FFB6 8F00             GOSBVL =RPLLIN        go Replace line
           000
155 0FFBD 500              RTNNC
156 0FFC0 8C00   =MfErr    GOLONG =MFERR
           00
157                *
```

```
158                *
159                *
160                ********************************************************************
161                ********************************************************************
162                **
163                ** Name:    LSTADR  -  Last Address on a Port
164                **
165                ** Category:   ADDCAL
166                **
167                ** Purpose:
168                **      Calculates last address on a port
169                **
170                ** Entry:
171                **      D(7-0) =  Nibs 1-8 of config table
172                **
173                ** Exit:
174                **      P      =  0
175                **      C(A)   =  last address on port (equiv. to AVMEME)
176                **      A(A)   =  #nibs on port
177                **      D   intact from entry
178                **
179                ** Calls:      MSIZE+
180                **
181                ** Uses.......
182                **  Exclusive: A, C
183                **  Inclusive: A, C, P
184                **
185                ** Stk lvls:   1
186                **
187                ** History:
188                **
189                **    Date      Programmer              Modification
190                **  --------   ----------   ----------------------------------
191                **  07/08/82   SW              Updated documentation
192                **
193                ********************************************************************
194                ********************************************************************
195                * ASSUMES MSIZE DOESN'T USE D
196                *
197 OFFC6 AFB  =LSTADR C=D    W
198 OFFC9 7000         GOSUB  =MSIZE+
199 OFFCD F0           ASL    A
200 OFFCF F0           ASL    A
201                * A(A) CONTAINS NUMBER OF NIBS ON PORT
202 OFFD1 AFB          C=D    W
203 OFFD4 BF6          CSR    W
204 OFFD7 AE2          C=0    B
205                * C(A) CONTAINS STARTING ADDRESS ON PORT
206 OFFDA C2           C=C+A  A              LAST ADDRESS (AVMEME)
207 OFFDC 03           RTNCC
208                *
```

```
209                 STITLE Find Line in Program File
210               ********************************************************************
211               ********************************************************************
212               **
213               ** Name:(S) FINDL   -  Find Line# within a Program File
214               **
215               ** Category:    FILUTL
216               **
217               ** Purpose:
218               **      Attempt to find passed in Line# within program and
219               **      Return with pointer to start of line
220               **
221               ** Entry:
222               **      FINDLR: Read Line# @ D0 into C(A)
223               **      FINDL : C(A) =   Line# to find
224               **      FINDL+: B(A) =   Line# to find
225               **      FINDLO: B(A) =   Line# to find
226               **              C(A) =   Start of Search
227               **              D(A) =   End of Search
228               **
229               **      Assumes: File type = BASIC
230               **
231               ** Exit:
232               **      D(A) = End of CURRENT file
233               **      D0   = Previous line found
234               **           = 0 if No previous line found
235               **
236               **      Carry set
237               **              Line# found
238               **              D1 @ Line#
239               **              S0=0, S1=0
240               **
241               **      Carry clear:
242               **              S1=1        --->   NULL program     - D1 past EOF
243               **              S0=1        --->   Line# not found  - D1 past EOF
244               **              S0=0,S1=0 --->   Line# > found    - D1 @ line#
245               **
246               **      If line# found      - Carry set
247               **                            D1 @ Line# found
248               **                            S0=0, S1=0
249               **      If line# > found    - Carry Clear
250               **                            D1 @ Line# > found
251               **                            S0=0, S1=0
252               **      If Null Program     - Carry clear
253               **                            D1 points past EOF on file
254               **                            S0=1, S1=1
255               **      If line# not found - Carry clear
256               **                            D1 points past EOF on file
257               **                            S0=1, S1=0
258               **      Error Exit -
259               **        None
260               **
261               ** Calls:    NULLP, NXTLIN
262               **
263               ** Uses.....
```

```
264                **  Exclusive: A(A),B(A),C(A),D(A),D0,D1,S0,S1
265                **  Inclusive: A(A),B(A),C(A),D(A),D0,D1,S0,S1
266                **
267                ** Stk lvls: +3
268                **
269                ** Detail:   NULLP
270                **              Carry Set if Null Program
271                **              D1= First line of file
272                **              D = End of current file
273                **              C = oBSsod
274                **    Assumes: When end of program test done in FINDL
275                **              If not null program, NOT @ end of program
276                **              C (00011) is ALWAYS < D (End of program)
277                **
278                ** History:
279                **
280                **    Date:     Programmer            Modification
281                **  --------    ----------    ---------------------------------
282                **  01/04/83    JP            Removed S9 usage
283                **  03/01/83    JP            Updated documentation
284                **                            NULLP does not Error Exit
285                **
286                ********************************************************************
287                ********************************************************************
288                *
289 0FFDE D2    =FINDLR C=0    A
290 0FFE0 15E3         C=DAT0 4              Read line# into C
291 0FFE4 D5    =FINDL  B=C    A
292 0FFE6 21    =FINDL+ P=     1              Don't want to check or rtn file typ
293 0FFE8 7660         GOSUB  NullP          Check if Null Program
294 0FFEC D0           A=0    A
295 0FFEE 130          D0=A                  Initial Previous Line#=0
296 0FFF1 443          GOC    FINDL7         Move past EOF, Set flag and return
297 0FFF4 171          D1=D1+ 2              Move past EOF
298 0FFF7 137          CD1EX
299 0FFFA 135          D1=C
300                *
301                * Assumes If not NULL program --> Not @ End of program
302                * Therefore, End of program test will ALWAYS be false (C=00010)
303                *
304 0FFFD 840   =FINDL0 ST=0   0              Clear LIne# > found flag
305 10000 841          ST=0   1              Clear Null Program flag
306 10003 8BF   FINDL2 ?C>=D  A              At end of program ?
307 10006 62           GOYES  FINDL8         Line # not found - Set S0,rtn
308 10008 15B3         A=DAT1 4              Read line# in program
309 1000C 8A0          ?A=B   A              Line #s match?
310 1000F 00           RTNYES                Line # found - Set carry on rtn
311 10011 8B0          ?A>B   A
312 10014 B1           GOYES  FINDL9         Line# > found - Clr carry & rtn
313 10016 133          AD1EX                 Save previous line in D0
314 10019 130          D0=A
315 1001C 131          D1=A                  Preserve D1
316 1001F 7E00         GOSUB  NXTLIN
317 10023 5FD          GONC   FINDL2         (B.E.T.)
318                *
```

```
319 10026 171   FINDL7 D1=D1+ 2          Positon past EOF
320 10029 851          ST=1   1          NULL PROG FLAG
321 1002C 850   FINDL8 ST=1   0          LINE# NOT FOUND
322 1002F 03    FINDL9 RTNCC              CLEAR CARRY ON RETURN
```

```
                        STITLE Scan to Next Line
                *****************************************************************
                *****************************************************************
                **
                ** Name:(S) NXTLIN  -  Scan to Next Line
                **
                ** Category:    FILUTL
                **
                ** Purpose:
                **      Scan from Line Number to End of Line Token
                **
                ** Entry:
                **      D1 @ Line Number
                **
                ** Exit:
                **      Carry Clear
                **      D1, C(A) POINT PAST EOL TOKEN
                **
                ** Calls:       None
                **
                ** Uses.......
                **  Exclusive: A(A),C(A),D1
                **  Inclusive: A(A),C(A),D1
                **
                ** Stk lvls:   +0
                **
                ** Detail:
                **      USES IMPLEMENTATION OF '@' FOR MULTI-STATEMENT LINES
                **
                *****************************************************************
                *****************************************************************
                *
                *
10031 173       =NXTLIN D1=D1+ 4              Move ptr to line length byte
10034 D0         NXTLI2 A=0    A
10036 14B              A=DAT1 B               Read line length
10039 137              CD1EX
1003C C2               C=C+A  A               Add line length to data pointer
1003E 135              D1=C
10041 14B              A=DAT1 B
10044 171              D1=D1+ 2
10047 E6               C=C+1  A
10049 E6               C=C+1  A
1004B 90C              ?A#0   P               NOT EOL?
1004E 6E               GOYES  NXTLI2          IF NOT EOL, GOTO NXTLI2
10050 01               RTN                    CARRY CLEAR ON RETURN
                *
                *
10052 8D00       NullP  GOVLNG =NULLP
      000
```

```
372                 STITLE Cursor Up and Down, Decompile Line
373               ********************************************************************
374               ********************************************************************
375               **
376               ** Name:(S) CURSRU  -  Cursor Up
377               ** Name:(S) CURSRD  -  Cursor Down
378               ** Name:(S) CURTOP  -  Cursor Top
379               ** Name:(S) CURBOT  -  Cursor Bottom
380               ** Name:(S) DCPLIN  -  Decompile line and display it
381               ** Name:(S) DSPLI+  -  Display line with cursor on;calc cursor po
382               ** Name:(S) DSPLIN  -  Display line with cursor on;pass cursor po
383               **
384               ** Category:   DCMUTL
385               **
386               ** Purpose:  Cursor UP, Cursor DOWN, Cursor TOP, Cursor BOTTOM
387               **             FETCH "next line" in program memory
388               **             Scroll Cursor Up | Cursor Down
389               **           Decompile and Display line with Cursor on
390               **
391               ** Entry:    CURBOT: Sets Cursor Bottom flag        sCURBT
392               **                   Clear Cursor Up flag          sCURUP
393               **                   Displays last line of non-null program
394               **           CURTOP: Clears Cursor Bottom flag      sCURBT
395               **                   Displays first line in a non-null program
396               **           CUR020: Entry for FETCH w/ CURRL=0
397               **                   Assumes sCURBT=0
398               **                   Avoids CR/LF and Poll for Cursor UP/DOWN
399               **           CURSRU: Sets Cursor Up flag            sCURUP
400               **           CURSRD: Clears Cursor Up flag          sCURUP
401               **           DCPLIN: Decompile & Display Line Entry
402               **                   D1 @ Line to decompile
403               **           DSPLI+: Display line entry in output buffer
404               **                   #cursor rights needed will be calculated
405               **                   FETCH KEY entry
406               **           DSPLIN: Display Line Entry in Output Buffer
407               **                   AUTOX entry
408               **                   A(A) = #backspaces for cursor position
409               **                        = #cursor rights
410               **                   OUTBS @ Start of line to decompile
411               **
412               **           CURRL = Current line# referenced
413               **           After FINDL call:
414               **                 S0=0 if Line# > found
415               **                 S0=1 if Line# not found
416               **                 S1=1 if Null program memory
417               **                 D0 = Previous Line found
418               **                 D =  End of current program
419               **
420               ** Exit:     If Private program
421               **              Error Exit <-- eFPROT
422               **           If not BASIC program
423               **              Poll on pCURSR
424               **              If no response:
425               **                 Error Exit <-- eFTYPE
426               **           If no CURRent Line # or Null Program file
```

```
427             **              Return to Main Loop
428             **            else
429             **              D1 = Start of line to Decompile
430             **              Decompile & Display line w/ Cursor
431             **              Return to MAIN30 to preserve display
432             **
433             ** Calls:     FINDL,LDCOMP,BF2DPP,DSPCHO,NXTLIN,RDCHDR
434             **            NULLP,BLDDSP,FPOLL,CURRLO,DO=OBS,CURSRT,
435             **            CRGTPR (CRLFSD & GETPeF)
436             **
437             ** Uses:      A-D,D0,D1,CURRL,R0-R2,S0,S1,S5-S8
438             **            For Cursor entry: sCURUP (S2), sCURBT (S3)
439             **
440             **              sCURUP = Cursor Up
441             **              sCURBT = Cursor Bottom
442             **              R0=  # backspaces for cursor position after line#
443             **
444             ** Stk Lvls:   5
445             **
446             ** Detail:
447             **
448             **   sCURBOT, sCURUP set/cleared for pCURSR to guarantee
449             **   unique determined of Cursor key.
450             **
451             **
452             **   CURBOT: Clear Cursor Up flag                    (sCURUP)
453             **           Set Cursor Bottom flag                  (sCURBT)
454             **           goto 0:
455             **   CURTOP: Clear Cursor Bottom flag                (sCURBT)
456             **           Set Cursor Up flag
457             **        0: Send Carriage Return / Line Feed        (CRGTPR)
458             **           If Private Program                      (GETPeF)
459             **              Error Exit                           (eFPROT)
460             **           Set status to check file type & error   (S9)
461             **           If non BASIC program                 (Carry set)
462             **      0.5:    Poll for Cursor keys                 (pCURSR)
463             **              Error exit if no response            (eFTYPE)
464             **   CUR020:  If NULL Program
465             **              golong MAINLP
466             **           If CURTOP
467             **              Position to line# of First line      (D1+1EOL)
468             **              go Decompile and Display line        (DCPLIN)
469             **           else
470             **              go Find Last Line in file
471             **                 Line# <--- FFFF                   (goto 3)
472             **   CURSRU: Set Cursor Up flag                      (sCURUP)
473             **           Set Cursor Bottom flag
474             **           goto 1;
475             **   CURSRD: Clear Cursor Up flag                    (sCURUP)
476             **           Clear Cursor Bottom flag
477             **        1: Send CR/LF                              (CRGTPR)
478             **           If Private program                      (GETPeF)
479             **              Error Exit                           (eFPROT)
480             **           If non-BASIC program                 (Carry set)
481             **              Issue pCURSR poll                    (goto 0.5)
```

```
482             **          Read current Line#                          (CURRLO)
483             **       3: Find Line#                                  (FINDL)
484             **          If Line# NOT found     (Carry clear)
485             **             If Cursor Down
486             **                If Line# > found                      (S0=0)
487             **                   go Decompile & Display line        (DCPLIN)
488             **                else
489             **                   If NULL program                    (S1=1)
490             **                      goto Main Loop
491             **                   else
492             **                      go Decompile previous line      (goto 4)
493             **             If Cursor Up                            (sCURUP)
494             **                If NULL program                       (S1=1)
495             **                   goto Main Loop
496             **                else
497             **                   Get First line of file             (RDHDR1)
498             **                   go Decompile previous|first line (goto 4)
499             **
500             **          If Line# found    (Carry set)
501             **             If Cursor Down
502             **                Save current line position          (B)
503             **                Get next line                       (NXTLIN)
504             **                If next line >= End of program
505             **                   Next line <-- Saved current line
506             **                go Decompile & display line         (DCPLIN)
507             **             If Cursor Up                           (sCURUP)
508             **       4:       If previous line # 0                (D0)
509             **                   Next line = Previous line
510             **                else
511             **                   D1 @ Line to Decompile & Display
512             **
513             **   DCPLIN: Decompile line @ D1                      (LDCOMP)
514             **           Calculate # backspaces to space after line#
515             **   DSPLIN: Display line
516             **              Send prompt                           (BF2DPP)
517             **              Send buffer                           (DSPCN0)
518             **              Send backspaces (cursor rights)       (CURSRT)
519             **              Build display                         (BLDDSP)
520             **           Return to Main, Keep display             (MAIN30)
521             **
522             ** History:
523             **
524             **    Date     Programmer     Modification
525             **  --------   ------------   --------------------------------
526             **  03/01/83   JP             Added pCURSR poll on File Type
527             **  04/12/83   JP             Ignore CURRL=0
528             **  04/12/83   JP             CUR020 entry point for FETCH
529             **  07/15/83   JP             Send CR/LF before Private check
530             **
531             ********************************************************************
532             ********************************************************************
533             ********************************************************************
534             ********************************************************************
535             **
536             ** Name:(S) pCURSR  -  Cursor Key with non BASIC file Poll
```

```
537             **
538             ** Category:   POLL
539             **
540             ** Type:       FPOLL
541             **
542             ** Purpose:
543             **      Fast Poll to allow the Cursor Keys to be used with
544             **        non BASIC files:
545             **      Cursor Up, Cursor Down, Cursor Top, Cursor Bottom
546             **
547             ** Should poll be "Handled" (return with XM=0)?:
548             **      No this is a TAKE OVER poll
549             **
550             ** Meaning of "Handling" Poll (what does code do if handled?):
551             **      Perform Cursor Key on file, return to MAIN30/MAINLP
552             **      See notes below.
553             **
554             ** Entry conditions for handler:
555             **      Carry set
556             **      B[A] = Poll number = pCURSR
557             **      HEX mode.
558             **      P=0.
559             **
560             **      Type of Key:     Status: sCURUP (S2)  sCURBT (S3)
561             **      ------------             ------       ------
562             **      Cursor Bottom              0            1
563             **      Cursor Top                 1            0
564             **      Cursor Up                  1            1
565             **      Cursor Down                0            0
566             **
567             **      Call RDCHD+ to get Filetype returned in R2
568             **
569             ** Normal exit conditions from handler if handled (ST, RAM,
570             ** registers, etc.):
571             **
572             **      HEX mode
573             **      Perform Cursor Key on file
574             **      GOVLNG to MAIN30
575             **
576             ** Normal exit conditions from handler if not handled (ST, RAM,
577             ** registers, etc.):
578             **      HEX mode.
579             **      XM=1.
580             **      S2 and S3 must be preserved
581             **
582             ** Available subroutine levels: 5
583             **      FPOLL handler is two levels deeper than caller
584             **        Invoked from CURSOR keys --- top level
585             **
586             ** NOTE:
587             **      The file type of the current file can be determined:
588             **          Call RDCHD+; R2 = File type on return
589             **
590             ** What registers/RAM may be used if handled?:
591             **      A-D, D0, D1, P always available--
```

```
592             **      Anything may be used: Status,R0-R4, SCRTCH...
593             **      CURRL (4 nibs) holds the current BASIC file line number
594             **        This field may be used if CURRENT file is line numbered
595             **
596             ** What registers/RAM may be used if not handled?:
597             **      A-C, D[15-5] D0, D1, P always available
598             **      NOTE: D[A] is sacred in FPOLL!!--
599             **      R0-R4
600             **
601             ** Special memory/pointer considerations (are pointers funny?):
602             **      Take care when returning to the MAIN LOOP
603             **
604             **      MAIN30 is the return point for Cursor Keys in BASIC
605             **        The line has been decompiled
606             **        The prompt is sent and the display built (BLDDSP)
607             **      MAINLP is the return point if NOTHING is displayed
608             **      CR/LF with no delay has been sent (S-CRLF) prior
609             **        to displaying the line.
610             **
611             **
612             ** Envisioned application(s):
613             **      Allow Cursor keys to display lines of a non BASIC file
614             **
615             **      The handler is responsible for maintaining the
616             **      "Current file" position.  Possibly an I/O Buffer can
617             **      be used.
618             **
619             ** History:
620             **
621             **    Date     Programmer             Modification
622             **  --------   ----------   ----------------------------------
623             **  03/01/83   JP           Added poll
624             **  04/14/83   JP           Revised documentation
625             **  06/02/83   SW           If null file, check for AUTO mode;
626             **                           Not AUTO mode => goto MAINLP
627             **                               AUTO mode => display curr line
628             **                          (Before, went to MAINLP regardless)
629             **
630             ************************************************************
631             ************************************************************
632             *
633             * Cursor BOTTOM Entry
634             *    Set Cursor Bottom flag
635             *    Clear Cursor Up flag
636             *
637 10059 853  =CURBOT ST=1   sCURBT         Set Cursor Bottom flag
638 1005C 842         ST=0   sCURUP         Clear Cursor Up flag
639 1005F 6900        GOTO   CUR010
640             *
641             * Cursor TOP Entry
642             *   Clear Cursor Bottom flag
643             *   Set Cursor Up flag for pCURSR
644             *     Never used within code
645             *
646 10063 843  =CURTOP ST=0   sCURBT         Clear Cursor Bottom flag
```

```
    647 10066 852          ST=1   sCURUP         Set for pCURSR
    648               *
    649               *   Send Carriage Return/Line Feed
    650               *   If Private program ---> Error exit from GETPeF
    651               *
    652 10069 79E0  CUR010 GOSUB  CRGTPR         Send CR/LF, Check protection
    653 1006D 521          GONC   CUR020         BASIC file ?
    654               *
    655               *   If non BASIC file
    656               *      POLL on pCURSR with non-BASIC file type
    657               *      Error Exit ---> Illegal File type
    658               *
    659 10070 8E00  CUR015 GOSUBL =FPOLL
              00
    660 10076 92           CON(2) =pCURSR        Cursor Up/Down with nonBASIC file
    661 10078 D2           C=0    A              If no response
    662 1007A 8C00         GOLONG =FTYPER           Illegal Filetype Error
              00
    663               *
    664               * FETCH entry:
    665               *    CURRL = 0 and not a null program
    666               *    Display first line of file
    667               * If Null Program
    668               *    Return to Main Loop
    669               *
    670 10080 7ECF =CUR020 GOSUB   NullP          Null Program ?
    671 10084 475          GOC    CUREXT         Yes - Return to Main Loop
    672               *
    673               * If CURTOP
    674               *    Position to first line of file  (@ EOL before from NULLP)
    675               *    go Decompile & Display line
    676               *
    677               * If CURBOT
    678               *    Set Line# = FFFF
    679               *    go Find Last line in file
    680               *
    681 10087 873   CUR030 ?ST=1  sCURBT         Cursor Bottom ?
    682 1008A 90           GOYES  CUR035
    683 1008C 171          D1=D1+ 1EOL           Position past EOL
    684 1008F 6870         GOTO   DCPLIN         Display First line
    685 10093 D2    CUR035 C=0    A
    686 10095 CE           C=C-1  A              Line# = FFFF
    687 10097 4D1          GOC    CUR060         B.E.T.
    688               *
    689               * Cursor Up Entry
    690               *    Set Cursor Up flag
    691               *    Set Cursor Bottom flag for pCURSR
    692               *      Never check within this code
    693               *
    694 1009A 852  =CURSRU ST=1   sCURUP         Cursor UP entry flag
    695 1009D 853          ST=1   sCURBT         Set for pCURSR
    696 100A0 6900         GOTO   CUR040
    697               *
    698               * Cursor Down Entry
    699               *    Clear Cursor Up flag
```

```
700             *    Clear Cursor Bottom flag for pCURSR
701             *      Never checked within this code
702             *
703 100A4 842  =CURSRD ST=0   sCURUP          Cursor DOWN entry
704 100A7 843          ST=0   sCURBT          Clear this for pCURSR
705             *
706             * Send CR/LF
707             * If Private program
708             *    Error Exit
709             * If non BASIC
710             *    go Poll, then Error
711             *
712 100AA 78A0  CUR040 GOSUB  CRGTPR
713 100AE 41C          GOC    CUR015          non BASIC program
714             *
715             * Read current line#
716             *
717 100B1 7CA0         GOSUB  CURRLO          Read CURRL, test for 0
718             *
719             * Find Current Line#
720             *
721 100B5 7B2F  CUR060 GOSUB  FINDL           Find line
722 100B9 5D1          GONC   CUR065          Line# not found
723             *
724             * Line# found:      Cursor Down
725             *
726             *    Save start of current line
727             *    Position to next line
728             *    If next line >= End of program file
729             *       Next line <-- Saved current line
730             *    go Decompile & Display line
731             *
732 100BC 872          ?ST=1  sCURUP          Cursor Up ?
733 100BF E3           GOYES  CUR110
734 100C1 137          CD1EX
735 100C4 D5           B=C    A               Save start of current line
736 100C6 137          CD1EX
737 100C9 746F         GOSUB  NXTLIN          Position to next line
738 100CD 8B3          ?C<D   A               Next line within program ?
739 100D0 83           GOYES  DCPLIN
740 100D2 D4           A=B    A               Next line = Saved current line
741 100D4 503          GONC   CUR115          B.E.T.
742             *
743             * Line# NOT found:
744             *    If Null program
745             *       Return to Main Loop
746             *    If Cursor DOWN
747             *       If line# > found
748             *          go Decompile and Display @ Line
749             *       else
750             *          Previous Line (D0) @ Last line of file
751             *
752 100D7 861   CUR065 ?ST=0  1               Not Null Program ?
753 100DA 90           GOYES  CUR070
754 100DC 8D00  CUREXT GOVLNG =CURSNP         Cursor on null program
```

```
          000
    755           *
    756 100E3 872  CUR070 ?ST=1  sCURUP         Cursor Up ?
    757 100E6 A0          GOYES  CUR090
    758 100E8 860         ?ST=0  0              Line# > found ?
    759 100EB D1          GOYES  DCPLIN         go Decompile/Display line
    760 100ED 5F0         GONC   CUR110         B.E.T.
    761           *
    762           * Line# NOT found:  Cursor Up
    763           *
    764           *    Get first line of file
    765           *    If previous line from FINDL = 0
    766           *       Decompile @ First line of file
    767           *    else
    768           *       Decompile @ Previous line
    769           *
    770 100F0 8F00 CUR090 GOSBVL =RDCHDR        Read current file header
          000
    771 100F7 17F         D1=D1+ 16             Kludge to keep symbolic
    772 100FA 170         D1=D1+ (oBSsod)-16    Position past first EOL of file
    773           *
    774           * If previous line # 0
    775           *    Next line = Previous line
    776           * else
    777           *    D1 @ Line to Decompile/Display
    778           *
    779 100FD 132  CUR110 ADOEX                 Previous line
    780 10100 8A8         ?A=0   A              No previous line ?
    781 10103 50          GOYES  DCPLIN
    782 10105 131  CUR115 D1=A                  Set line @ Previous 1
    783           *
    784           * DCPLIN - Decompile Line @ D1
    785           *
    786 10108 8F00 =DCPLIN GOSBVL =LDCOMP       Decompile Line
          000
    787 1010F 8E00 =DSPLI+ GOSUBL =d0=obs       C(A) pts to OUTPUT bufst
          00
    788           *
    789           * Calc #spaces to move cursor right
    790           *
    791 10115 1F1C         D1=(5) =LDCSPC       Addr @ space after Line#
          6F2
    792 1011C AD0          A=0    M
    793 1011F 143          A=DAT1 A
    794 10122 EA           A=A-C  A             Difference
    795 10124 81C          ASRB                 Divide by 2
    796           * A(A) >=1
    797           * DSPLIN - Display Line
    798           *
    799 10127 100  =DSPLIN R0=A                 Save # spaces
    800 1012A D9           C=B    A
    801           *
    802           * NOTE:
    803           *   BF2DSP uses 3 levels
    804           *   Call to it uses 4 levels
```

```
805             *   5 levels used to save # characters
806             *   All entries to this routine are GOVLNG
807             *
808 1012C 06            RSTK=C                  Save # characters
809             *
810             * Send Prompt, Buffer, # cursor rights, Build display
811             *
812 1012E 8F00          GOSBVL =BF2DPP          Display PROMPT
          000
813 10135 07            C=RSTK                  Restore # characters to B(A)
814 10137 D5            B=C    A
815 10139 8E00          GOSUBL =DSPCNO          Display Output Buffer
          00
816 1013F 118           C=R0                    #cursor rights
817 10142 8E00          GOSUBL =CURSRT          CURSFL, then move cursor right
          00
818 10148 8F00          GOSBVL =BLDDSP          Build display
          000
819 1014F 8D00  CUR130  GOVLNG =MAIN30          Return to Main, keep display
          000
820             *
821             * Subroutine:  Send CR/LF with no delay
822             *              Check file protection
823             *               Error exit if Private from GETPeF
824             *               Return carry set if non BASIC
825 10156 7000  CRGTPR GOSUB  =crlfsd           Send CR/LF w/ no delay
826 1015A 8D00          GOVLNG =GETPeF          Check private/file type
          000
```

```
827                     STITLE Read and Test CURRL for zero
828                 ****************************************************************
829                 ****************************************************************
830                 **
831                 ** Name:    CURRLO  -  Read and Test CURRL for Zero
832                 **
833                 ** Category:   GENUTL
834                 **
835                 ** Purpose:
836                 **      Read current line and test if current line = 0
837                 **
838                 ** Entry:
839                 **
840                 **
841                 ** Exit:
842                 **      P untouched
843                 **      C(A) = Current Line  (CURRL)
844                 **      Carry clear
845                 **        C(A) = 0 = Current line
846                 **      Carry set
847                 **        C(A) # 0 = Current line
848                 **
849                 ** Calls:      None
850                 **
851                 ** Uses.......
852                 **  Exclusive: C(A),D0
853                 **
854                 ** Stk lvls:   0
855                 **
856                 ** NOTE:
857                 **      D1 could be used instead of D0
858                 **
859                 ** History:
860                 **
861                 **    Date     Programmer              Modification
862                 **  --------   ----------   ----------------------------------
863                 **  10/08/82   JP           Added routine
864                 **  10/21/82   JP           Changed sense of Carry on return
865                 **
866                 ****************************************************************
867                 ****************************************************************
868 10161 1B8E =CURRLO D0=(5) =CURRL
        7F2
869 10168 D2            C=0    A              Zero C(4)
870 1016A 15E3          C=DAT0 4              Read current line#
871 1016E 8AE           ?C#0   A              Current line# # 0 ?
872 10171 00            RTNYES                Carry set if Current line# # 0
873 10173 01            RTN                   Carry clear if Current Line# = 0
```

```
                        STITLE Save Line# in CURRL
                *****************************************************************
                *****************************************************************
                **
                ** Name:    SAVEL#  -  Update Current Line# (CURRL)
                **
                ** Category:   FILUTL
                **
                ** Purpose:
                **      Update current line# RAM location: CURRL
                **
                ** Entry:
                **      SAVEL+:    C(A) @ Start of line with new line#
                **      SAVEL#:    D1   @ Start of line with new line#
                **      SAVELO:    C(A) = New line #
                **
                ** Exit:
                **      Carry clear
                **      D0 @ CURRL RAM location
                **      SAVEL+:     CURRL = Line # at C
                **                          D1 @ Start of line#
                **      SAVEL#:     CURRL = Line # at D1
                **      SAVELO:     CURRL = Line # in C
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Exclusive: C(A),D0,CURRL
                **  Inclusive: C(A),D0,CURRL
                **
                ** Stk lvls:   +0
                **
                *****************************************************************
                *****************************************************************
                *
10175 135       =SAVEL+ D1=C
10178 D2        =SAVEL# C=0     A
1017A 15F3              C=DAT1 4                Read in Line# at D1
1017E 1B8E      =SAVELO D0=(5) =CURRL           Write C into CURRL
      7F2
10185 15C3              DAT0=C 4
10189 03                RTNCC
```

```
915                     STITLE Send CR/LF
916                ********************************************************************
917                ********************************************************************
918                **
919                ** Name:    SCRLF>  -  Send Carriage Return/Line Feed
920                **
921                ** Category:   LOCAL
922                **
923                ** Purpose:
924                **      Send Carriage Return and Line Feed with delay suppressed
925                **
926                ** Entry:
927                **      SCRLF>: Sends CR,LF with delay suppressed
928                **
929                ** Exit:
930                **      CR and LF sent to Display Buffer
931                **      Display is nulled
932                **
933                **      Exit through BF2DSP
934                **
935                ** Calls:      none
936                **
937                ** Stk lvls:   +3
938                **
939                ** Uses.......
940                **  Exclusive: D1
941                **  Inclusive: D1,same as BF2DSP
942                **
943                ********************************************************************
944                ********************************************************************
945                *
946 1018B 8D00  SCRLF> GOVLNG =CRLFND
          000
947                *
948 10192               END
```

BF2DPP	Ext		-	812					
BLDDSP	Ext		-	818					
CHKPSF	Ext		-	103					
CLLINK	Ext		-	114					
CLPSTK	Ext		-	104					
CRGTPR	Abs	65878	#10156 -	825	652	712			
CRLFND	Ext		-	946					
CUR010	Abs	65641	#10069 -	652	639				
CUR015	Abs	65648	#10070 -	659	713				
=CUR020	Abs	65664	#10080 -	670	653				
CUR030	Abs	65671	#10087 -	681					
CUR035	Abs	65683	#10093 -	685	682				
CUR040	Abs	65706	#100AA -	712	696				
CUR060	Abs	65717	#100B5 -	721	687				
CUR065	Abs	65751	#100D7 -	752	722				
CUR070	Abs	65763	#100E3 -	756	753				
CUR090	Abs	65776	#100F0 -	770	757				
CUR110	Abs	65789	#100FD -	779	733	760			
CUR115	Abs	65797	#10105 -	782	741				
CUR130	Abs	65871	#1014F -	819					
=CURBOT	Abs	65625	#10059 -	637					
CUREXT	Abs	65756	#100DC -	754	671				
CURRL	Abs	194536	#2F7E8 -	11	868	912			
=CURRLO	Abs	65889	#10161 -	868	717				
CURSNP	Ext		-	754					
=CURSRD	Abs	65700	#100A4 -	703					
CURSRT	Ext		-	817					
=CURSRU	Abs	65690	#1009A -	694					
=CURTOP	Abs	65635	#10063 -	646					
DOOUTB	Ext		-	122					
D1=CRS	Ext		-	153					
=DCPLIN	Abs	65800	#10108 -	786	684	739	759	781	
DSPCNO	Ext		-	815					
=DSPLI+	Abs	65807	#1010F -	787					
=DSPLIN	Abs	65831	#10127 -	799					
=FINDL	Abs	65508	#0FFE4 -	291	721				
=FINDL+	Abs	65510	#0FFE6 -	292	128				
=FINDLO	Abs	65533	#0FFFD -	304					
FINDL2	Abs	65539	#10003 -	306	317				
FINDL7	Abs	65574	#10026 -	319	296				
FINDL8	Abs	65580	#1002C -	321	307				
FINDL9	Abs	65583	#1002F -	322	312				
=FINDLR	Abs	65502	#0FFDE -	289					
FPOLL	Ext		-	659					
FTYPER	Ext		-	662					
GETPeF	Ext		-	826					
LDCOMP	Ext		-	786					
LDCSPC	Abs	194241	#2F6C1 -	11	791				
=LSTADR	Abs	65478	#0FFC6 -	197					
MAIN30	Ext		-	819					
MFERR	Ext		-	156					
MSIZE+	Ext		-	198					
=MfErr	Abs	65472	#0FFC0 -	156					
NULLP	Ext		-	371					
NXTLI2	Abs	65588	#10034 -	357	367				

=NXTLIN	Abs	65585	#10031	-	356	141	316	737		
NullP	Abs	65618	#10052	-	371	293	670			
OBCOLL	Ext			-	127					
=PEDIT	Abs	65375	#0FF5F	-	93					
PEDIT0	Abs	65392	#0FF70	-	114					
PEDIT2	Abs	65424	#0FF90	-	128	126				
PEDIT3	Abs	65447	#0FFA7	-	150	137				
=PEDITD	Abs	65378	#0FF62	-	102					
=PEDITM	Abs	65392	#0FF70	-	113					
PEDT00	Abs	65378	#0FF62	-	103					
RDCHDR	Ext			-	770					
RPLLIN	Ext			-	154					
=SAVEL#	Abs	65912	#10178	-	910					
=SAVEL+	Abs	65909	#10175	-	909	123				
=SAVELO	Abs	65918	#1017E	-	912					
SCRLF>	Abs	65931	#1018B	-	946					
crlfsd	Ext			-	825					
d0=obs	Ext			-	787					
lEOL	Abs	2	#00002	-	12	683				
oBSsod	Abs	17	#00011	-	12	772				
pCURSR	Abs	41	#00029	-	12	660				
sCURBT	Abs	3	#00003	-	12	637	646	681	695	704
sCURUP	Abs	2	#00002	-	12	638	647	694	703	732	756

Input Parameters

  Source file name is JP&MEM::MS

  Listing file name is JP/MEM:TI:ML::-1

  Object file name is JP%MEM:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *   M   M  N   N   &      CCC   N   N  FFFFF
 2            *   MM MM  N   N  & &    C   C  N   N  F
 3            *   M M M  NN  N  & &    C      NN  N  F
 4            *   M M M  N N N   &     C      N N N  FFFF
 5            *   M   M  N  NN  & & &  C      N  NN  F
 6            *   M   M  N   N  &  &   C   C  N   N  F
 7            *   M   M  N   N   && &   CCC   N   N  F
 8            *
 9                 TITLE  Configuration Module <831213.1255>
10 10192          ABS    #10192
11                RDSYMB SB%RAM::MS
12                RDSYMB TI%EQU::MS
```

```
                        STITLE ISRAM? utility
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:(S) ISRAM?  -  Pointing At RAM?
                    **
                    ** Category:   CONFIG
                    **
                    ** Purpose:
                    **      Determine whether an address is in RAM or something
                    **      else.  This was put in to save writing to non-RAM
                    **      devices, which for ROMs does no harm but for EEPROMs
                    **      does plenty of harm.
                    **
                    ** Entry:
                    **      Address to check in C[A].
                    **
                    ** Exit:
                    **      P=0.
                    **      Carry set if address is in system RAM or IRAM.
                    **      Address passed is now in B[A].
                    **
                    ** Calls:      CNFFND, MSIZ++
                    **
                    ** Uses.......
                    **              A,B[A],C,D1.
                    **
                    ** Stk lvls:   1
                    **
                    ** History:
                    **
                    **    Date      Programmer              Modification
                    **  --------    ----------   ----------------------------------
                    **  12/09/82    NM           Wrote.
                    **
                    ****************************************************************
                    ****************************************************************
10192 D5    =ISRAM? B=C     A               Hold address.
10194 20            P=      0
10196 3400          LC(5)   =INTR4          Start of RAM.
      4F2
1019D 8B5           ?B<C    A               Address before start?
101A0 D6            GOYES   ISRMCC          Yes.  RTNCC.
101A2 73E2          GOSUB   C=RAME          C=(RAMEND)
101A6 8B5           ?B<C    A               Address before end?
101A9 00            RTNYES                  Yes.  RTNSC.
101AB 31EF          LCHEX   FE
101AF 8E7F          GOSUBL  CNFFND          Look for ROM buffer.
      70
101B5 500           RTNNC                   RTNCC if not found.
101B8 171           D1=D1+  2               For looking at table entries.
101BB D2     ISRM10 C=0     A
101BD 30A           LC(1)   10
101C0 EA            A=A-C   A               Decrement counter.
101C2 4A4           GOC     ISRMCC          RTNCC if no more.
```

```
 66 101C5 15F6          C=DAT1 7          Read nibs 2-8 of entry.
 67 101C9 AE2           C=0    B          Address of this module.
 68 101CC E1            B=B-C  A          Offset of addr rel to this mod.
 69 101CE 402           GOC    ISRM20     Go if addr before module.
 70 101D1 14F           C=DAT1 B          Restore size information.
 71 101D4 8E00          GOSUBL =ASLW3     Move counter out of way.
          00
 72 101DA 7F22          GOSUB  MSIZ++     Compute size of module.
 73 101DE D6            C=A    A
 74 101E0 F2            CSL    A
 75 101E2 F2            CSL    A          Size in nibs.
 76 101E4 8E00          GOSUBL =ASRW3     Restore counter.
          00
 77 101EA 8B9           ?C<=B  A          Set carry if addr not in module.
 78 101ED 20            GOYES  ISRM20
 79 101EF 15F5  ISRM20  C=DAT1 6
 80 101F3 AE2           C=0    B          Fetch address again.
 81 101F6 5B0           GONC   ISRM30     Go if found module.
 82 101F9 C1            B=B+C  A          Restore address.
 83 101FB 179           D1=D1+ 10         Point at next entry.
 84 101FE 6CBF          GOTO   ISRM10
 85 10202 C1    ISRM30  B=B+C  A          Restore address.
 86 10204 25            P=     5          Point at device type.
 87 10206 A0E           C=C-1  P          Decrement device type.
 88 10209 20            P=     0
 89 1020B 01            RTN               Carry set if device RAM.
 90 1020D 03    ISRMCC  RTNCC
```

```
                STITLE Hardware Configuration code
*****************************************************************
*****************************************************************
**
** Name:(S) CONF    -  Configure Everything
**
** Category:   CONFIG
**
** Purpose:
**      To configure all soft-configurable devices on the
**      system bus.
**
** Entry:
**  CONF: S0=0 if requesting a power-up configuration
**             (preserve integrity of system),
**           1 if requesting a coldstart configuration (reset
**             all pointers to coldstart values).
**  CONFS3: S0 as above plus:
**          S3 = 1 if we intentionally want configuration to
**               behave as though ROM configuration changed.
**
** Exit:
**      Configuration proper falls through to LEXBUF.  S0
**      indicates whether a power-up configuration (S0=0) or
**      a coldstart configuration (S0=1) was done.
**
** Calls:       AD1P,C=MAIN,C=RAME,CDIV10,CLKSPD,CLRXDS,
**              CONFP4,CSLC3,CSLW4,CSLW5,CSRC3,CSRW3,
**              D=AVME,DSLW-P,FNDBUB,INITPT,MODSIZ,MOVED2,
**              MOVEUA,MOved3,MRKNEW,MRKOLD,MSIZE,Moveu3,
**              R3<RST,RFADJ+,ROMTPT,RST<R3,SIZE10,SORT,
**              SORTP2,STMBF?,TBLPT+,TBLPTR,UNCFG8,WAITKY,
**              WHLTBL,csrw6.
**
** Uses.......
**              A,B,C,D,D0,D1,P,R0-R4,Display buffer,S0-S3,
**              RSTKBF.
**
** Stk lvls:   3 (four are saved in RSTKBF)
**
** NOTE:
**      The configuration code may decide on its own to perform
**      a coldstart configuration when a power-up config was
**      requested.  This would be done if certain memory was
**      corrupt, disallowing the manipulations necessary to
**      maintain system integrity.  In this case, the code
**      will GOVLNG to COLDST (address #00000), which will
**      wipe out the machine and call this code with S0=1.
**
**      If configuration code determines that ROM configuration
**      has changed to a point endangering the validity of
**      the umpteen pointers in the mainframe, it will
**      essentially perform an "EDIT workfile" before falling
**      into the LEXBUF code.  It will also close all files
**      in the FIB.  This may be forced by entering at CONFS3
```

```
146             **      with S3=1.
147             **
148             **      If code detects the presence of too many ROMS to
149             **      configure in the address space, it will give a warning
150             **      message.  It is not written to cover the contingency
151             **      of too many RAMS or MMI/O devices, on the assumption
152             **      that the possibility of said happening is too small
153             **      to merit the immense code required.
154             **
155             ** Detail:
156             **
157             ** This code configures all soft-configurable devices on the
158             ** system Bus.  The code builds three tables in the configu-
159             ** ration buffers: System RAM, Other memory (ROM, EEPROM,
160             ** independent RAM, etc.), Memory-mapped I/O.  The buffer IDs
161             ** for the above configuration tables are, respectively,
162             ** FF, FE, FD.  The exact format of the information in the
163             ** tables is explained below.
164             **
165             **
166             ** Following is the pre-configuration memory layout:
167             **
168             **   00000-1FFFF: Operating system
169             **   2C000-2C01F: Card reader
170             **   2E100-2E3FF: Display RAM
171             **   2F400-2FFFF: Disp Driver RAM
172             **   (FFC00-FFFFF: Reserved for config garbage collection)
173             **
174             ** The configuration code assigns addresses as follows:
175             **
176             ** Memory-mapped I/O upward from 20000-28000.
177             **
178             ** System RAM contiguously upward from 30000.
179             **   To achieve this contiguous mapping, system RAM is
180             **   configured in reverse size order.  This assures that
181             **   64 Knib RAMS are configured on 64 Knib boundaries,
182             **   32 Knib RAMS on 32 Knib boundaries, etc.
183             **
184             ** Uses S0-S3 internally as follows:
185             **      S0: Set for coldstart, clear for power-up configure.
186             **      S1: Used internally in debubbling system RAM, then
187             **            used to indicate presence of ROMs for which
188             **            there is no room to configure.  Results in
189             **            message.
190             **          (Debubbling is explained in algorithm (below)
191             **            shortly below CON400 label.)
192             **      S2: Set to indicate failure of internal file chain
193             **            verify.  Results in message.
194             **      S3: Set to indicate that system ROM configuration has
195             **            changed to an extent which may endanger the
196             **            validity of some pointers.  Results in collapsing
197             **            of stacks and resetting pointers as though an
198             **            "edit" command was entered.
199             **
200             ** To explain configuration, the following terms are used
```

```
201             ** below:
202             **
203             **     PORT#: Physical port location (1-5) whose daisy chain
204             **            is addressed by a bit (0-4) in output register.
205             **            Port #0 is the internal daisy chain.
206             **            Port #5 goes to the card reader slot.
207             **     DEV#: Position of a plug-in (0-15) in a daisy chain.
208             **           Unless there is a port extender, all plug-ins
209             **           will be device #0.
210             **     SEQUENCE: Consecutive chips in a module to be used
211             **               as a single entity (e.g., a quad RAM which
212             **               appears as one plug-in to the user).
213             **     DEVICE TYPE: Type of memory (RAM, ROM, etc., or memory-
214             **                  mapped I/O).
215             **     DEVICE CLASS: Identifies memory-mapped I/O device.
216             **
217             **
218             ** *** CHIP ID ***
219             **
220             **   The CHIP ID is a (usually) mask-programmed 20-bit pattern
221             **   which is read by the CPU on an ID poll (C=ID instruction).
222             **   A chip responds to the ID poll if two conditions are met:
223             **     1) The chip is unconfigured,
224             **     2) Daisy-in is high on the chip.
225             **   By examining the daisy chains one at a time, configuring
226             **   each chip as we find it, we can locate and identify all
227             **   soft-configurable chips on the bus.
228             **
229             ** The chip-id contains the following information:
230             **
231             **   NIBBLE 0: 15-Log2(size).
232             **       Memory Size        Nib 0     MM I/O space
233             **       -----------        -----     ------------
234             **        1 knib              F         1 word  (16 nibs)
235             **        2                   E         2
236             **        4                   D         4
237             **        8                   C         8
238             **        16                  B         16
239             **        32                  A         32
240             **        64   (max RAM)     9         64
241             **        128                 8         128
242             **        256  (max memory)  7         256
243             **                            6         512
244             **                            5         1024
245             **
246             **   NIBBLE 1: (Reserved for future use)
247             **             This nibble from the first chip in a
248             **             sequence is stored in the configuration
249             **             table for all sequences.
250             **
251             **   NIBBLE 2: Device type-- 0: RAM
252             **                           1: ROM
253             **                           2-E: assorted memory types
254             **                           F: Memory-mapped I/O
255             **
```

```
256           **   NIBBLE 3: For memory, (unassigned).
257           **             For memory-mapped I/O, contains device
258           **               class--     0: HPIL mailbox
259           **                           1-15: (unassigned)
260           **             (Note: Card reader is hard configured
261           **                    at 2C000-2C01F.)
262           **
263           **   NIBBLE 4: bits 0-1: (unassigned)
264           **             bit 2: Last chip in sequence (see note (1)
265           **                    below).
266           **                    Always assumed high for MM I/O
267           **                    devices, meaning all such devices
268           **                    have their own table entry.
269           **             bit 3: Last chip in module.
270           **
271           ** The top two bits (bits 2-3 of nibble 4) are used to
272           ** determine what chips are in what physical plug-ins.
273           ** Every sequence of chips (e.g., four identical RAMS in
274           ** a RAM plug-in, an applications pack containing two
275           ** ROMS, etc.) results in one entry in the configuration
276           ** tables.
277           **
278           ** (1) End of sequence (but not module) is identified in
279           ** one of two ways: 1) next chip returns ID with different
280           ** value in nibs 0-3;  2) last chip of sequence has bit
281           ** 18 set.  The second approach is necessary if
282           ** consecutive, identical chips are to be considered as
283           ** different sequences, and will probably NEVER be used
284           ** in the entire lifetime of the machine.  But it can
285           ** be done.
286           **
287           ** A module containing four 8-Kbit RAMS might return the
288           ** following sequence of IDs:
289           **         0000E  0000E  0000E  8000E
290           ** The resulting table entry would identify the chip
291           ** size, chip count, device type, physical location,
292           ** and configuration address of the device.
293           **
294           ** A module containing two 128-Kbit ROMS, a memory-mapped
295           ** I/O interface using 2 words of address space, and four
296           ** 16-Kbit RAMS might present the following sequence of
297           ** IDs:
298           **         0010A First ROM           \ one ROMtable entry
299           **         0010A End of ROM sequence /
300           **         01F0E MM I/O devclass 1   one MM I/Otable entry
301           **         0000D Start of RAMS       \
302           **         0000D                     | one RAMtable entry
303           **         0000D                     |
304           **         8000D End of module       /
305           **
306           ** Restrictions: 16 chips/sequence
307           **               16 sequences/device
308           **               16 devices/port
309           **
310           **
```

```
311             ** Format of table entries:
312             **
313             **          System RAM              Other Memory
314             **          (cnftable ID FF)        (cnftable ID FE)
315             **          ----------              ------------
316             ** NIB 0    Seq position            Seq position
317             ** NIB 1    Device #                Device #
318             ** NIB 2    Port #                  Port #
319             ** NIB 3    15-Log2(size) **        15-Log2(size)
320             ** NIB 4 /                         /
321             ** NIB 5 | Address (kbit)         | Address (kbit)
322             ** NIB 6 \                         \
323             ** NIB 7    0                       Device type
324             ** NIB 8    #chips/plugin-1         #chips/plugin-1
325             ** NIB 9    Nibble 1 from ID        Nibble 1 from ID
326             **
327             **          Memory-mapped I/O
328             **          (cnftable ID FD)
329             **          ----------------
330             ** NIB 0    Sequence position in dev
331             ** NIB 1    Device #
332             ** NIB 2    Port #
333             ** NIB 3    15-Log2(size)
334             ** NIB 4 /
335             ** NIB 5 | Address (words rel to 10000)
336             ** NIB 6 \
337             ** NIB 7    Device type  (always F)
338             ** NIB 8    Device class
339             ** NIB 9    Nibble 1 from ID
340             **
341             ** ** FREEPORT routine may set this to zero to indicate
342             **    that the RAM has been removed intentionally.  This
343             **    affects operation of this code in the spot where the
344             **    old and new tables are compared to determine which
345             **    RAMs are new and which are missing.
346             **
347             ** Algorithm:
348             **   CONF:
349             **      S3=0 {to indicate we do not want EDITWF unless
350             **        necessary}.
351             **   CONFS3:
352             **      Save 4 subroutine levels in RSTKBF.
353             **   CONFRS:
354             **      B=0000000000000001 {B contains device counters and
355             **        other good things: B[B]=bit for output register,
356             **        B[XS]=device#, B[3]=port#, B[S]=sequence#, B[6-5]=
357             **        RAM counter, B[8-7]=ROM counter, B[10-9]=MMIO
358             **        counter, B[12-11]=sum of other three counters,
359             **        B[4]=(temporary storage of ID hinib).}
360             **      D1=start of display buffer area where we build table.
361             **      Perform a bus reset.
362             **   IDLOOP:
363             **      Is there room for any more entries?  If not then goto
364             **        CONF10.
365             **      Energize daisy chain for this port (OUT=B[B]/2).
```

```
366             **      Get ID of next device on daisy chain (C=ID).
367             **      If response#0 then goto IDLP20.
368             **      Increment port# (B[3]).
369             **      Reset device# (B[XS]).
370             **      Reset sequence# (B[S]).
371             **      Move port bit over one (B=B+B B).
372             **      If port bit<=80H then goto IDLOOP else goto CONF10.
373             **  IDLP20:
374             **      Hold ID in R3.
375             **      Hold ID hinibble in B[4].
376             **      Build device table entry (except address) in C.
377             **      If devicetype=RAM then goto IDLP90.
378             **      If devicetype#Memory-mapped-I/O then goto IDLP60.
379             **      Write memory-mapped-I/O table entry at D1.  Configure
380             **        device to 40000H.
381             **      P=(position of MMIO counter).
382             **  IDLP30:
383             **      If hibit of ID clear then goto IDLP40.
384             **      Increment device#.
385             **      Reset sequence#.
386             **  IDLP40:
387             **      Increment device counter pointed to by P.
388             **      Increment total-#-devices counter.
389             **      Goto IDLOOP.
390             **  IDLP60:  {configuring "ROMs"}
391             **      Set address field of table entry to FFF00.
392             **      Find and configure all chips in this sequence to
393             **        40000H (gosub CONFP4).
394             **      P=(position of ROM counter).
395             **      Goto IDLP30.
396             **  IDLP90:  {configuring RAMs}
397             **      Configure chip to 80000H.
398             **      If first 8 nibbles of chip = IRAM ID then unconfigure
399             **        chip and goto IDLP60 {if IRAM then treat as ROM}.
400             **      Unconfigure chip.
401             **      Find and configure all chips in this sequence to
402             **        400000H (gosub CONFP4).
403             **      P=(position of RAM counter).
404             **      Goto IDLP30.
405             **  CONF10: {Having identified everything plugged in...}
406             **      Sort table by device type. {Sorting on WP, where P=7.
407             **        Besides separating RAMs from ROMs from MMI/O, this
408             **        will separate RAMs from IRAMs, since IRAMs were given
409             **        an address (FFF00) while RAMs were not, and address
410             **        serves as a secondary sort key.}  This will arrange
411             **        table into three pieces: RAM, ROM, MMIO.
412             **      A=300H {starting address/100H of first RAM}.
413             **  CONF20: {assign addresses to RAM table entries}
414             **      If there are no more system RAMs in table then goto
415             **        CONF60.
416             **      Write A[X] to address field of table entry.
417             **      Increment A[X] by module size {module size=chipsize *
418             **        #chips in module}.
419             **      Goto CONF20.
420             **  CONF60:
```

```
421             **      Save RAMEND {A[X]*100H} in R1.
422             **      Point at ROM table.  Sort it by size.
423             **      Clear B for building allocation map {B will contain a
424             **        bitmap of what pages --a page is 10000H nibbles-- are
425             **        available for configuring ROMs}.
426             **      If there is anything non-zero at E0000 {i.e., a hard-
427             **        configured device} then B[15]=B[14]=F {mark those
428             **        pages as unavailable}.
429             **      Mark pages as unavailable which are occupied or
430             **        partially occupied by operating system and system
431             **        RAM.
432             **  CONF70: {loop to assign addresses to big ROMs}
433             **      Any more ROMs in table?  If not then goto CON170.
434             **      If size of this entry < 1 page then goto PAKROM.
435             **      Compute legal configuration boundaries and # pages
436             **        needed for this ROM.
437             **      Examine bitmap (starting at high end) for possible
438             **        locations to configure this ROM.
439             **      If possible location is found, write address to table
440             **        entry.  {Otherwise, table entry still contains FFC00
441             **        from ID loop}.  Mark allocation map for space taken
442             **        by this ROM.
443             **      Goto CONF70.
444             **  PAKROM: {loop to assign addresses to small ROMs}
445             **      Compute boundaries of one or (if available) two
446             **        bubbles (blobs of unconfigured address space).
447             **  PAKR50:
448             **      Examine ROM table entry;
449             **        If ROM fits in either bubble, write address to
450             **        table entry and reduce bubblesize appropriately.
451             **      If there are more ROMs in list then goto PAKR50.
452             **  CON170: {now to configure memory-mapped I/O}
453             **      A=0 X (address of MM I/O relative to 20000H).
454             **  CON180:
455             **      If no more table entries then goto CON210.
456             **      Write A[X] to table entry.
457             **      Add device size to A[X].
458             **      Goto CON180.
459             **  CON210:
460             **      Sort entire table (RAMs, ROMs and MMI/O) by port-dev#.
461             **      Perform bus reset.
462             **  CON220: {loop to configure all at assigned addresses}
463             **      Any more table entries?  If not then goto CON270.
464             **      Read table entry.
465             **      Compute output register value for this port.  OUT=C.
466             **      If not memory-mapped I/O then goto CON240.
467             **      Compute configuration address (20000H + [addr]*10H)
468             **        and issue CONFIG command at that address.
469             **      Goto CON220.
470             **  CON240:
471             **      Compute chipsize (from table entry) and configuration
472             **        address ([table entry]*100H).
473             **      Configure all chips in the sequence contiguously.  If
474             **        address=FFF00, then do not increment address for each
475             **        chip {this is rubbish plug-in, to be unconfigured
```

```
476             **        soon; all chips goto FFF00}.
477             **      Unconfigure everything at FFF00 {chips for which there
478             **        wasn't room}.
479             **      If R4[A] has been disturbed since we began {an
480             **        interrupt occurred, and the output register may have
481             **        been screwed} then goto CONFRS {start over}.
482             **      Sort entire table by device type {separates system RAMs
483             **        from "ROMs" from MMIO}.
484             **      Sort RAM table by port-device# {for comparison with old
485             **        table in configuration buffers}.
486             **      {Time for the hard work.  If this is a coldstart we
487             **        will initialize all system pointers.  If this is not
488             **        we need to compare the old and new RAMtables and move
489             **        memory to adjust for any modules which may have been
490             **        added since the last configuration.}
491             **      Was coldstart requested on entry (S0=1)?  If not then
492             **        goto CON280.
493             **    coldst:  {here if config decides to coldstart}
494             **      Was coldstart requested on entry?  If not then GOVLNG
495             **        to 00000.
496             **      Clear password.
497             **      Initialize all pointers, filechain, command stack,
498             **        variable chain heads, I/O buffers to coldstart
499             **        values.
500             **      Goto PUTBUF.
501             **    CON280: {ready to incorporate new RAMs}
502             **      Look for old RAMtable.  If not there then goto coldst.
503             **    CON310: {start of loop to compare RAM tables}
504             **      Anything more in newtable?  If not then goto CON380.
505             **      Anything more in oldtable?  If not then goto CON390.
506             **    CON330:
507             **      Read two table entries.  If size, port-dev#, sequence#
508             **        and chipcount the same then goto CON310.
509             **      If newtable pdev# < oldtable pdev# {newtable has new
510             **        device} then goto CON360.
511             **      If newtable pdev# > oldtable pdev# {oldtable has
512             **        missing device} then goto CON350.
513             **      {Pdev#'s the same.  Something went away, something else
514             **        appeared.}
515             **      Mark newtable entry as new.
516             **      Mark oldtable entry as missing.
517             **      Goto CON310.
518             **    CON350:
519             **      Mark oldtable entry as missing.
520             **      Goto CON310.
521             **    CON360:
522             **      Mark newtable entry as new.
523             **      Increment newtable pointer.
524             **      If more newtable entries goto CON330.
525             **    CON370: {remaining oldtable entries missing}
526             **      Mark oldtable entry as missing.
527             **    CON380:
528             **      Any more oldtable entries?  If yes then goto CON370
529             **        else goto CON400.
530             **    CON390: {remaining newtable entries new}
```

```
                **      Mark newtable entry as missing.
                **      Any more entries?  If yes then goto CON390.
                **  CON400:
                **      Read current values of AVMEMS and AVMEME.  Look at
                **        oldtable.  If any entries are marked as missing and
                **        were not entirely contained between AVMEMS and AVMEME
                **        then goto coldst.
                **      Compute nw AVMEME.  Store in R3.
                **      {Now comes the really hard part.  We will rearrange
                **        everything in memory to restore contiguity in light
                **        of any system RAMs which were added.}
                **      Sort RAMtable by address.
                **      Point past last entry in RAM table {we well read
                **        back from end of table}.
                **      {This is a debubbling process; that is, removing
                **        "bubbles" of new memory from existing memory.  This
                **        is done by creating a zero-length bubble at RAMEND.
                **        The bubble is then moved down through memory, passing
                **        RAMs which are not marked as new and expanding to
                **        contain RAMs which are marked as new.  This process
                **        continues until the bubble hits available memory,
                **        which is itself a bubble.}
                **      D0=new RAMEND {lowbound of bubble}.
                **      R3=new RAMEND {highbound of bubble}.
                **      D=old AVMEME {to know when we are done}.
                **      S1=0 {indicate that we are not almost done}.
                **  CON470:
                **      Any more table entries?  If yes then goto CON480.
                **      Dope up entry to look like built-in hard-configured
                **        RAM.
                **      S1=1 {indicate we are almost done}.
                **      Goto CON490.
                **  CON480:
                **      Read next table entry down.
                **      Marked as new?  If not then goto CON490.
                **      D0=D0-modulesize {expand bubble by changing lowbound}.
                **      Goto CON470.
                **  CON490:
                **      If bubblesize#0 then goto CON500.
                **      D0=D0-modulesize {move lowerbound of bubble}.
                **      R3=R3-modulesize {move upperbound of bubble}.
                **      If S1=1 {i.e., if we are almost done} then goto CON550
                **        else goto CON470.
                **  CON500:
                **      Move bubble down (i.e., move data up) size of module.
                **        If there is nothing to move (i.e., we have hit
                **        available memory) then goto CON550.
                **      If S1=0 goto CON470.
                **  CON550:
                **      {now that we have debubbled the stacks, it is time to
                **        debubble program memory.}
                **      R3=30000H {lowbound of bubble}.
                **      D0=30000H {highbound of bubble}.
                **      D=AVMEMS {to determine when we are done}.
                **  CON560: {start of loop}
```

```
586           **      Any more table entries?  If not then goto CON650.
587           **      Read next entry.  If not marked as new then goto
588           **        CON580.
589           **      Increase upperbound of bubble (D0) by size of this
590           **        module.
591           **      Goto CON560.
592           **  CON580:
593           **      Move bubble down past this module (i.e., move that
594           **        amount of data down--to lower memory).
595           **      If we are not done (bubble has not hit available
596           **        memory) then goto CON560.
597           **  CON650:
598           **      Unmark all RAMtable entries which were marked as new.
599           **      Update all pointers past AVMEME {since available memory
600           **        may have changed size}.
601           **      Update variable chain heads.
602           **      Sort RAMtable in port-dev# order {since this will be
603           **        the oldtable next time, it needs to be in this
604           **        order).
605           **      Sort ROMtable in port-dev# order.
606           **      Look for old ROMtable.  If not found then goto coldst.
607           **      Compare old and new ROMtables.  If any old ROMs are
608           **        missing or moved then S3=1 {indicate that we wish to
609           **        force an edit-workfile to occur}.
610           **  PUTBUF:
611           **      Sort Memory-mapped I/O table by port-dev#.
612           **      Delete all table entries in all tables with an assigned
613           **        address of FFF00 {these were not configured}.  If any
614           **        entries deleted, S1=1 {indicate that configuration
615           **        error has occurred}.
616           **      {Now we will move tables from display buffer, where
617           **        they were built, to configuration buffer area, where
618           **        they will live, and will be known as oldtables on the
619           **        next configuration.}
620           **      Compute size needed for configuration tables.  Compute
621           **        size taken by current configuration tables.  Compute
622           **        difference and move memory to make proper amount of
623           **        room.
624           **      If there is insufficient memory to hold new tables,
625           **        pinch off tables one entry at a time until there is
626           **        room and indicate configuration error {S1=1}.
627           **      Move tables from display buffer to configuration buffer
628           **        area.
629           **      Compute clockspeed and store in RAM (gosbvl CLKSPD).
630           **      Restore subroutine levels saved at beginning.
631           **      Fall through to LXBF++.
632           **      { Configuration proper is done.  The LXBF code will
633           **        find and build tables of all lexfiles.  It will also
634           **        report configuration error if that was requested
635           **        and perform an edit-workfile if that was requested.
636           **        That could not be done at this point in the code
637           **        because some polls are issued, and that cannot be
638           **        done until we have a valid list of lexfiles.}
639           **
640           ** History:
```

```
641             **
642             **    Date      Programmer            Modification
643             **  --------    ----------   ---------------------------------
644             **  09/15/82    NM           Added name to documentation
645             **
646             ********************************************************************
647             ********************************************************************
648 1020F 840   =PWCONF ST=0   0              Request powerup configuration
649 10212 843   =CONF   ST=0   3              Indicate config has not changed
650 10215 04    =CONFS3 SETHEX                B[2]=dev#,b[3]=prt#,b[b]=port bit
651 10217 77C7          GOSUB  R3<RST         Save 4 subroutine levels
652 1021B D2    CONFRS  C=0    A
653 1021D A6E           C=C-1  B              C[A]=000FF
654 10220 10C           R4=C                  Will remain untouched if no intr
655 10223 808F          INTOFF
656 10227 AF1           B=0    W              Counters: RAM[6-5],rom[8-7],
657 1022A E5            B=B+1  A              MMIO[10-9],seqpos[15],sum[12-11]
658 1022C 1FB7          D1=(5) =ESCSTA        Start of table
          4F2
659 10233 80A           RESET                 Unconfigure all chips
660 10236 133   IDLOOP  AD1EX
661 10239 20            P=     0
662 1023B 34E4          LC(5)  (=DSPMSK)+24-10
          5F2
663 10242 8B6           ?A>C   A              Can we fit more entries?
664 10245 B2            GOYES  IDLP10         No
665 10247 133           AD1EX                 Yes
666 1024A D2            C=0    A
667 1024C AE9           C=B    B
668 1024F 81E           CSRB
669 10252 801           OUT=C                 Energize daisy chain
670 10255 77D0          GOSUB  WAITKY         Wait for key up.
671 10259 806           C=ID                  Anything out there?
672 1025C 96E           ?C#0   B
673 1025F 51            GOYES  IDLP20         Yes
674 10261 B55           B=B+1  M              No.  increment port#
675 10264 AA1           B=0    XS             Reset device#
676 10267 AC1           B=0    S              Reset seqctr
677 1026A A65           B=B+B  B              Move port bit
678 1026D 58C           GONC   IDLOOP         Go if more ports
679 10270 6D22  IDLP10  GOTO   CONF10         Done w/ID loop.  Jump past utils.
680 10274 DA    IDLP20  A=C    A              Copy id
681 10276 10B           R3=C                  Hold in r3
682 10279 24            P=     4
683 1027B A85           B=C    P              Copy hinib of id to b[4]
684 1027E 20            P=     0
685 10280 80F1          CPEX   1
686 10284 80F4          CPEX   4              Will hold res'd nibble in c[9]
687 10288 8E00          GOSUBL =CSLW4         Type,class to 6-7
          00
688 1028E 23            P=     3
689 10290 A99           C=B    WP
690 10293 F6            CSR    A
691 10295 F6            CSR    A              Size,port,dev to nibs 0-2
692 10297 AC9           C=B    S              Copy sequence position
```

```
693 1029A 812          CSLC
694 1029D 928          ?A=0   XS            Is this RAM?
695 102A0 05           GOYES  IDLP90        Yes, check if independent
696 102A2 B24          A=A+1  XS            No, is this MM i/o?
697 102A5 593          GONC   IDLP60        No, this is other memory
698 102A8 15D9         DAT1=C 10            Yes, write out table entry
699 102AC 179          D1=D1+ 10
700 102AF D2           C=0    A
701 102B1 24           P=     4
702 102B3 304          LCHEX  4             40000
703 102B6 7670         GOSUB  WAITKY        Wait for key up.
704 102BA 805          CONFIG               Configure device
705 102BD A05          B=B+B  P             Hibit set?
706 102C0 29           P=     9             Point to devtype counter
707 102C2 5B0  IDLP30  GONC   IDLP40        Go if hibit clear
708 102C5 B25          B=B+1  XS            Increment device counter
709 102C8 AC1          B=0    S
710 102CB A4D          B=B-1  S             Reset sequence position
711 102CE B45  IDLP40  B=B+1  S             Increment sequence position
712 102D1 7B60         GOSUB  AD1P          Increment devtype counter
713 102D5 2B           P=     11
714 102D7 7560         GOSUB  AD1P          Update total #devices counter
715 102DB 6A5F         GOTO   IDLOOP
716 102DF 24   IDLP60  P=     4             Other memory
717 102E1 32FF         LCHEX  FFF           Address =FFF00 for now
            F
718 102E6 7270         GOSUB  CONFP4        Configure module to 40000
719 102EA 27           P=     7             Point at devtype counter
720 102EC 65DF         GOTO   IDLP30
721 102F0 AF7  IDLP90  D=C    W             RAM, hold table entry in d
722 102F3 D2           C=0    A
723 102F5 24           P=     4
724 102F7 308          LCHEX  8             80000
725 102FA 7230         GOSUB  WAITKY        Wait for key down.
726 102FE 805          CONFIG
727 10301 134          DO=C
728 10304 15A7         A=DATO 8
729 10308 20           P=     0
730 1030A 37B3         LCHEX  EDDDDD3B      Independent RAM id
            DDDD
            DE
731 10314 27           P=     7
732 10316 912          ?A=C   WP            Match?
733 10319 01           GOYES  IDL100         Yes, process standalone RAM
734 1031B 7E20         GOSUB  UNCFG8         UNCNFG @ 80000 & rcovr tbl entry
735 1031F 7930         GOSUB  CONFP4         Configure module to 40000
736 10323 25           P=     5             Point to devtype counter
737 10325 6C9F         GOTO   IDLP30        Make table entry
738 10329      IDL100
739 10329 7020         GOSUB  UNCFG8         UNCNFG @ 80000 & rcovr tbl entry
740 1032D 51B          GONC   IDLP60        B.E.T.
```

```
                        STITLE Some of the utilities
                ********************************************************************
                ********************************************************************
                **
                ** Name:    WAITKY  -  Wait For Key Up
                **
                ** Category:   KEYUTL
                **
                ** Purpose:
                **      Wait for key up in energized rows.
                **
                ** Entry:
                **      Rows energized that we care about.
                **
                ** Exit:
                **      Carry set.
                **
                ** Calls:      KEYDN?
                **
                ** Uses.......
                **              Nothing.
                **
                ** Stk lvls:   2
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------   ----------   ----------------------------------
                **  10/12/82   NM           Wrote
                **
                ********************************************************************
                ********************************************************************
10330 7500      =WAITKY GOSUB  KEYDN?
10334 5BF               GONC   WAITKY         Key down, go back
10337 01                RTN
10339 8D00       KEYDN? GOVLNG =KYDN?
      000
                ********************************************************************
                ********************************************************************
                **
                ** Name:    AD1P    -  Increment Counter In B-reg Subfld
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Increment a 2-nibble counter in B pointed at by P.
                **
                ** Entry:
                **      P points at low nib of counter to increment.
                **
                ** Exit:
                **      Carry clear.
                **
                ** Calls:      None.
                **
```

```
795                ** Uses.......
796                **              P,B
797                **
798                ** Stk lvls:    0
799                **
800                ** History:
801                **
802                **    Date      Programmer              Modification
803                **  --------   ----------   ------------------------------------
804                **  10/11/82   NM           Wrote.
805                **
806                ********************************************************************
807                ********************************************************************
808 10340 B05    AD1P   B=B+1  P               Increment low digit.
809 10343 500           RTNNC                 Return if wasn't F.
810 10346 0C            P=P+1
811 10348 B05           B=B+1  P               Increment high digit.
812 1034B 03            RTNCC
813                ********************************************************************
814                ********************************************************************
815                **
816                ** Name:    UNCFG8  -  Unconfigure Chip At 80000.
817                **
818                ** Category:   LOCAL
819                **
820                ** Purpose:
821                **      Unconfigure a chip at 80000 and C=D.
822                **
823                ** Entry:
824                **
825                ** Exit:
826                **      Carry clear.
827                **
828                ** Calls:      None.
829                **
830                ** Uses.......
831                **              P,C.
832                **
833                ** Stk lvls:    0
834                **
835                ** History:
836                **
837                **    Date      Programmer              Modification
838                **  --------   ----------   ------------------------------------
839                **  10/11/82   NM           Wrote.
840                **
841                ********************************************************************
842                ********************************************************************
843 1034D D2     UNCFG8 C=0     A
844 1034F 24            P=      4
845 10351 308           LCHEX   8
846 10354 804           UNCNFG
847 10357 AFB           C=D     W
848 1035A 03            RTNCC
849                ********************************************************************
```

```
850             ****************************************************************
851             **
852             ** Name:    CONFP4  -  Configure A ROM/RAM Sequence
853             **
854             ** Category:   LOCAL
855             **
856             ** Purpose:
857             **      Configure a sequence of which main code has found the
858             **      first chip.
859             **
860             ** Entry:
861             **      C contains nibs 0-3, 7 and 9 from table ID (table ID
862             **        minus address and chipcount).
863             **      D1 is pointing at next table entry in the table being
864             **        built in display buffer area.
865             **      B[4] contains hinib of ID of first chip.
866             **      R3 contains ID of first chip (which has not yet been
867             **        configured).
868             **
869             ** Exit:
870             **      C contains nibs 0-3, 7-9 from table ID (table ID minus
871             **        address).
872             **      This almost-complete table entry has been written
873             **        to the table.
874             **      All chips in the sequence have been configured.
875             **      D1=D1+10.
876             **      P=4.
877             **      Carry set iff this is last sequence in module (hibit
878             **        of ID was set).
879             **
880             ** Calls:      KEYDN?
881             **
882             ** Uses.......
883             **  Exclusive: A,B[4],C,D,D1,P
884             **  Inclusive:
885             **
886             ** Stk lvls:   2
887             **
888             ** Detail:
889             **      This code is called to configure a seqence of RAM,
890             **      IRAM, or other memory.
891             **
892             ** History:
893             **
894             **    Date      Programmer              Modification
895             **  --------   ----------   ------------------------------------
896             **  06/21/82   NM           Added documentation
897             **
898             ****************************************************************
899             ****************************************************************
900 1035C 28    CONFP4 P=     8
901 1035E A82         C=0     P               Zero chipcounter
902 10361 AF7         D=C     W
903 10364 113         A=R3                    Id of first chip
904 10367 D2    CONP10 C=0     A
```

```
905 10369 24          P=     4
906 1036B 304         LCHEX  4              Configure address
907 1036E 77CF CONP15 GOSUB  KEYDN?         Key down?
908 10372 5BF         GONC   CONP15         Yes.  loop until up.
909 10375 805         CONFIG
910 10378 989         ?B>=C  P              Last chip in sequence or module?
911 1037B 12          GOYES  CONP20         Yes
912 1037D 78BF CONP17 GOSUB  KEYDN?         Key down?
913 10381 5BF         GONC   CONP17         Yes.  loop until up.
914 10384 806         C=ID                  Next chip
915 10387 23          P=     3
916 10389 916         ?A#C   WP             Same low 4 nibs as first chip id?
917 1038C 01          GOYES  CONP20         No; end of sequence
918 1038E 24          P=     4
919 10390 A85         B=C    P              Copy hinib
920 10393 28          P=     8
921 10395 B07         D=D+1  P              Increment chipcount
922 10398 6ECF        GOTO   CONP10
923 1039C AFB  CONP20 C=D    W
924 1039F 15D9        DAT1=C 10
925 103A3 179         D1=D1+ 10
926 103A6 24          P=     4
927 103A8 A05         B=B+B  P
928 103AB 01          RTN
929           ********************************************************************
930           ********************************************************************
931           **
932           ** Name:    SORT    -  Sort Table Entries
933           **
934           ** Category:   LOCAL
935           **
936           ** Purpose:
937           **      Sort entries in the configuration table.
938           **
939           ** Entry:
940           **      D1 points at start of table to be sorted.
941           **      C[B] = # entries.
942           **      P identifies sort field (sorts on WP).
943           **
944           ** Exit:
945           **      D1 points past end of table just sorted.
946           **
947           ** Calls:      None.
948           **
949           ** Uses.......
950           **  Exclusive: A,B,C,D0,D1
951           **
952           ** Stk lvls:   0
953           **
954           ** Detail:
955           **      Garden variety selection sort used often.  For example:
956           **      Sort entire table into Port-Device order for configur-
957           **        ing
958           **      Sort RAMtable (a subtable within the entire table) into
959           **        address order for restoring system integrity.
```

```
                        **      Sort tables into reverse size order for assigning
                        **        addresses.
                        **
                        ** History:
                        **
                        **    Date     Programmer          Modification
                        **  --------   ----------   ---------------------------------
                        **  06/21/82   NM           Added documentation
                        **
                        ********************************************************************
                        ********************************************************************
 103AD 22     SORTP2 P=     2
 103AF A6E    SORT   C=C-1  B             Outer loop
 103B2 400           RTNC
 103B5 D5            B=C    A
 103B7 F1            BSL    A
 103B9 F1            BSL    A             Outer loop counter to b[3-2]
 103BB AE5           B=C    B             Inner loop counter to b[b]
 103BE 137           CD1EX
 103C1 135           D1=C
 103C4 134           D0=C
 103C7 15B9          A=DAT1 10            Read first entry
 103CB 169    SORT20 D0=D0+ 10            Inner loop
 103CE A6D           B=B-1  B             Any more comparisons?
 103D1 451           GOC    SORT40        Go if no
 103D4 15E9          C=DAT0 10
 103D8 99A           ?A<=C  WP
 103DB OF            GOYES  SORT20
 103DD 1589          DAT0=A 10            A>c, swap
 103E1 AFA           A=C    W             Select new entry
 103E4 56E           GONC   SORT20        Examine rest of list.  B.E.T.
 103E7 1599   SORT40 DAT1=A 10            End inner loop
 103EB F5            BSR    A
 103ED F5            BSR    A             Recover outer loop counter
 103EF D9            C=B    A
 103F1 179           D1=D1+ 10
 103F4 6ABF          GOTO   SORT
                        ********************************************************************
                        ********************************************************************
                        **
                        ** Name:    TBLPTR  -  Point At Table
                        **
                        ** Category:   LOCAL
                        **
                        ** Purpose:
                        **      Point at table.
                        **
                        ** Entry:
                        **      TBLPTR: R0 points at something we want to point at.
                        **
                        ** Exit:
                        **      D1 = R0[A].
                        **      C = R0 SR'd 5 times (table entry counts in byte-size
                        **        chunks starting at C[0]).
                        **      D = C.
```

```
1015              **
1016              ** Calls:      CSRW5.
1017              **
1018              ** Uses.......
1019              **  Exclusive: D1 (first 2 entry points), C,D.
1020              **
1021              ** Stk lvls:   1
1022              **
1023              ** History:
1024              **
1025              **    Date      Programmer              Modification
1026              **  --------    ----------   ------------------------------------
1027              **  06/21/82    NM           Added documentation
1028              **
1029              ********************************************************************
1030              ********************************************************************
1031 103F8 118    TBLPTR C=R0                  Table pointer & counters
1032 103FB 135    TBLPT+ D1=C                  Point at table
1033 103FE 7690          GOSUB  csrw5
1034 10402 AF7           D=C    W
1035 10405 01            RTN
1036              ********************************************************************
1037              ********************************************************************
1038              **
1039              ** Name:    MSIZE   -  Compute Module Size
1040              **
1041              ** Category:   ADDCAL
1042              **
1043              ** Purpose:
1044              **      Compute the amount of address space occupied by a
1045              **      module.
1046              **
1047              ** Entry:
1048              **    MSIZE:
1049              **       C[3] = size nibble (nibble 3) from table entry.
1050              **       C[8] = chipcount nibble (nibble 8) from table entry.
1051              **       (Actually, C=table entry, but these are the only
1052              **         nibbles we use from it.)
1053              **    MSIZE+:
1054              **       C[2] = size nibble (nibble 3) from table entry.
1055              **       C[7] = chipcount nibble (nibble 8) from table entry.
1056              **       (i.e., table entry SR'd once.)
1057              **
1058              ** Exit:
1059              **       A[X]=(Module size in nibs)/100H.
1060              **       P=0.
1061              **       Carry set.
1062              **
1063              ** Calls:      None.
1064              **
1065              ** Uses.......
1066              **  Exclusive: A[X],C,P.
1067              **
1068              ** Stk lvls:   0
1069              **
```

```
1070            ** Algorithm:
1071            **      Convert chipsize nibble [15 - log2(chipsize)] to
1072            **        actual chipsize.
1073            **      Multiply by #chips in module.
1074            **
1075            ** History:
1076            **
1077            **    Date     Programmer              Modification
1078            **  --------   ----------   ----------------------------------
1079            **  06/21/82   NM           Added documentation
1080            **
1081            ********************************************************************
1082            ********************************************************************
1083 10407 BF6  =MSIZE  CSR    W
1084 1040A BF6  =MSIZE+ CSR    W
1085 1040D BF6  =MSIZ++ CSR    W
1086 10410 AB0   SIZE10 A=0    X
1087            ********************************************************************
1088            ********************************************************************
1089            **
1090            ** Name:    MODSIZ  -  Add Module Size To A
1091            **
1092            ** Category:   LOCAL
1093            **
1094            ** Purpose:
1095            **      Compute module size and add to A[X].
1096            **
1097            ** Entry:
1098            **      C[0]=nibble 3 of table entry (chipsize nibble).
1099            **      C[5]=nibble 8 of table entry (chipcount-1).
1100            **        (i.e., table entry SR'd 3 times).
1101            **      A[X]=number to add module size to.
1102            **
1103            ** Exit:
1104            **      A[X]=old A[X] + module size.
1105            **      P=0.
1106            **      Carry set.
1107            **
1108            ** Calls:      None.
1109            **
1110            ** Uses.......
1111            **  Exclusive: A[X],C,P.
1112            **
1113            ** Stk lvls:   0
1114            **
1115            ** History:
1116            **
1117            **    Date     Programmer              Modification
1118            **  --------   ----------   ----------------------------------
1119            **  06/21/82   NM           Added documentation
1120            **
1121            ********************************************************************
1122            ********************************************************************
1123 10413 80D0  MODSIZ P=C    0
1124 10417 0D            P=P-1                  14-log2(size)
```

```
1125 10419 AB2          C=0    X
1126 1041C E6           C=C+1  A
1127 1041E A36   MODS10 C=C+C  X              Loop to compute chipsize
1128 10421 0C           P=P+1
1129 10423 5AF          GONC   MODS10
1130 10426 25           P=     5              Point at #modules-1
1131 10428 A3A   MODS20 A=A+C  X              Add chipsize to a
1132 1042B A0E          C=C-1  P              Loop for # modules
1133 1042E 59F          GONC   MODS20
1134 10431 20           P=     0
1135 10433 02           RTNSC
1136                ****************************************************************
1137                ****************************************************************
1138                **
1139                ** Name:    ROMTPT  -  Point At Next Table
1140                **
1141                ** Category:   LOCAL
1142                **
1143                ** Purpose:
1144                **      Skip pointers past a table.
1145                **
1146                ** Entry:
1147                **      Pointer to table in D1.
1148                **      Table size (#entries) in D[B].
1149                **
1150                ** Exit:
1151                **      D1 points past end of table.
1152                **      D[B] SR'd twice.
1153                **      C[B]=D[B].
1154                **
1155                ** Calls:      None.
1156                **
1157                ** Uses.......
1158                **  Exclusive: A[A],C[A],D,D1
1159                **
1160                ** Stk lvls:   0
1161                **
1162                ** Detail:
1163                **      Typically D contains a series of table sizes (e.g.,
1164                **      D[B]=RAMtable size, D[3-2]=ROMtable size, D[5-4]= MMIO
1165                **      table size).
1166                **
1167                ** History:
1168                **
1169                **    Date      Programmer             Modification
1170                **  --------    ----------   ----------------------------------
1171                **  06/21/82    NM           Added documentation
1172                **
1173                ****************************************************************
1174                ****************************************************************
1175 10435 D2    ROMTPT C=0    A
1176 10437 AEB          C=D    B               #entries in RAMtable
1177 1043A BF7          DSR    W
1178 1043D BF7          DSR    W
1179 10440 133          AD1EX
```

```
1180 10443 C6          C=C+C  A              #*2
1181 10445 CA          A=A+C  A
1182 10447 C6          C=C+C  A
1183 10449 C6          C=C+C  A              #*8
1184 1044B CA          A=A+C  A
1185 1044D 133         AD1EX                 Point past RAMtable
1186 10450 DB          C=D    A
1187 10452 01          RTN
1188            ****************************************************************
1189            ****************************************************************
1190            **
1191            ** Name:    WHLTBL  -  Sort Entire Table (RAMS, ROMS, Etc.)
1192            **
1193            ** Category:   LOCAL
1194            **
1195            ** Purpose:
1196            **      Sort all three tables together on WP field.
1197            **
1198            ** Entry:
1199            **      P set to sort field.
1200            **      R0[A] points at start of table.
1201            **      R0[12-11] = #entries in all tables combined.
1202            **
1203            ** Exit:
1204            **      P unchanged.
1205            **      D[B]=# entries in all tables (# just sorted).
1206            **
1207            ** Calls:      TBLPTR, CSRW5, SORT
1208            **
1209            ** Uses.......
1210            **              A,B,C,D,D0,D1.
1211            **
1212            ** Stk lvls:   2
1213            **
1214            ** History:
1215            **
1216            **    Date      Programmer              Modification
1217            **  --------    ----------   ----------------------------------
1218            **  10/21/82    NM           Wrote.
1219            **
1220            ****************************************************************
1221            ****************************************************************
1222 10454 70AF  WHLTBL GOSUB  TBLPTR         Point at table & position ctrs.
1223 10458 7930        GOSUB  csrw6           Shift in #entries counter
1224 1045C D7          D=C    A               Copy to D.
1225 1045E 605F        GOTO   SORT            Sort it.
1226            ****************************************************************
1227            ****************************************************************
1228            **
1229            ** Name:    FNDBUB  -  Find Bubble Boundaries
1230            **
1231            ** Category:   LOCAL
1232            **
1233            ** Purpose:
1234            **      Look for bubble in address space.
```

```
1235                ** 
1236                ** Entry:
1237                **      P = page # where we begin search.
1238                **      B contains map of address space:
1239                **        0 in nib x if page x is vacant,
1240                **        F in nib x if page x is occupied.
1241                ** 
1242                ** Exit:
1243                **      C[X] is lower boundary of bubble / 100H.
1244                **      C[5-3] is upper boundary of bubble / 100H.
1245                **      (If C[X]=C[5-3]=000, there is no bubble.)
1246                **      If carry set, there are no more bubbles (we've hit end
1247                **        of address space).
1248                **      If carry clear, there may be more bubbles and P is
1249                **        positioned to continue the search for the next call.
1250                ** 
1251                ** Calls:       None.
1252                ** 
1253                ** Uses.......
1254                **             P,C[5-0].
1255                ** 
1256                ** Stk lvls:   None.
1257                ** 
1258                ** History:
1259                ** 
1260                **    Date      Programmer              Modification
1261                **  --------    ----------   -----------------------------------
1262                **  05/16/83    NM           Wrote
1263                ** 
1264                ********************************************************************
1265                ********************************************************************
1266 10462 0C     FNDBUB P=P+1                  Look for bottom of bubble
1267 10464 400          RTNC                   Return if no bubbles
1268 10467 90D          ?B#0   P               Found bottom of bubble?
1269 1046A 8F           GOYES  FNDBUB          No.
1270 1046C 80C2         C=P    2               Yes.  C[X]=address
1271 10470 0C     FNDB20 P=P+1                  Look for top of bubble
1272 10472 4D0          GOC    LCFFC           Go if end of addr space
1273 10475 909          ?B=0   P               Found top of bubble?
1274 10478 8F           GOYES  FNDB20          No.
1275 1047A 80C5         C=P    5               Yes.  C[5-3]=address
1276 1047E 03           RTNCC                  Ptr ready for next search.
1277 10480 23     LCFFC  P=     3               Here if hit end of addr space.
1278 10482 32CF         LCHEX  FFC             Top of last possible bubble.
          F
1279 10487 02           RTNSC
1280                ********************************************************************
1281                ********************************************************************
1282                ** 
1283                ** Name:    C=RAME  -  Read In RAMEND
1284                ** 
1285                ** Category:   GENUTL
1286                ** 
1287                ** Purpose:
1288                **      Read (RAMEND) into C.
```

```
1289                **
1290                ** Entry:
1291                **
1292                ** Exit:
1293                **      D1=RAMEND
1294                **      C[A]=(RAMEND)
1295                **      Carry unaffected.
1296                **
1297                ** Calls:      None.
1298                **
1299                ** Uses.......
1300                **             D1,C[A]
1301                **
1302                ** Stk lvls:   0
1303                **
1304                ** History:
1305                **
1306                **    Date     Programmer              Modification
1307                **  --------   ----------   ----------------------------------
1308                **  06/08/83   NM           Installed
1309                **
1310                ********************************************************************
1311                ********************************************************************
1312 10489 1F2B 5F2 =C=RAME D1=(5) =RAMEND
1313 10490 147          C=DAT1 A
1314 10493 01           RTN
1315                *
1316 10495 BF6   csrw6  CSR    W
1317 10498 8C00 00  csrw5  GOLONG =CSRW5
```

```
1318                        STITLE Configuration code (cont.)
1319 1049E AF9   CONF10 C=B    W              Devtype counters
1320 104A1 20           P=     0
1321 104A3 34B7         LC(5)  =ESCSTA        Start of table
           4F2
1322 104AA 108          R0=C                  Save tablepointer and counters
1323 104AD 27           P=     7
1324               *
1325               * Sort whole table by devtype.
1326               * Since address is secondary sort key and IRAMs have been
1327               *   assigned rubbish addresses, they will appear after
1328               *   RAMs (conveniently falling in the ROM table).
1329               * Since size is tertiary sort key, each device type will
1330               *   be sorted by size.  This is useful for RAMs and MMIO.
1331               *   Since ROMs encompass several device types, a subsequent
1332               *   sort by size will be needed.
1333               *
1334 104AF 71AF           GOSUB  WHLTBL         Sort whole table by devtype
1335 104B3 714F           GOSUB  TBLPTR         Point at table
1336 104B7 20           P=     0
1337 104B9 3200         LCHEX  300            30000 starting addr
           3
1338 104BE ABA          A=C    X
1339 104C1 172          D1=D1+ 3              Point at size
1340 104C4 A6F   CONF20 D=D-1  B              Any more RAMs?
1341 104C7 4B1          GOC    CONF60         No.
1342 104CA 15F5         C=DAT1 6              Yes, read nibs [8-3] of entry
1343 104CE 170          D1=D1+ 1
1344 104D1 1592         DAT1=A 3              Write out address
1345 104D5 178          D1=D1+ 9
1346 104D8 773F         GOSUB  MODSIZ         Add module size to address
1347 104DC 47E          GOC    CONF20         Next RAM.  B.E.T.
1348 104DF 6F41  CONF55 GOTO   CON170         Inrange shortjump from below
1349 104E3 F0    CONF60 ASL    A
1350 104E5 F0           ASL    A
1351 104E7 101          R1=A                  Save endmem
1352 104EA 7A0F         GOSUB  TBLPTR
1353 104EE 734F         GOSUB  ROMTPT
1354 104F2 23           P=     3
1355               *
1356               * Sort ROMs by size to assign addresses.
1357               *
1358 104F4 77BE         GOSUB  SORT           Sort ROMtbl by size
1359 104F8 7CFE         GOSUB  TBLPTR
1360 104FC 753F         GOSUB  ROMTPT         Point at ROMtable
1361 10500 172          D1=D1+ 3              Point at nib 3
1362 10503 AF1          B=0    W              Room for allocation map
1363               *
1364               * Check for presence of debugger.
1365               *
1366 10506 1B00         D0=(5) #E0000         Point at expected addr for dbugr
           00E
1367 1050D 1567         C=DAT0 W
1368 10511 97A          ?C=0   W              Anything there?
1369 10514 A0           GOYES  CONF69         No.
```

```
1370 10516 2E          P=     14            Yes.
1371 10518 A0D         B=B-1  P             Mark page E as taken.
1372 1051B A4D         B=B-1  S             Mark page F as taken.
1373             *
1374             * Note: Debugger won't work too well if RAMEND > E0000.
1375             *       Nor will machine, if debugger is present under
1376             *       said circumstances.
1377             *
1378 1051E 119  CONF69 C=R1                  Get RAMEND
1379 10521 CE          C=C-1  A
1380 10523 80D4        P=C    4
1381 10527 A1D         B=B-1  WP            Indicate pages taken for sys RAM
1382 1052A A6F  CONF70 D=D-1  B             Any more ROMs?
1383 1052D 41B         GOC    CONF55        No.  goto con170
1384 10530 15B5        A=DAT1 6             Nibs [8-3] of ROM table entry
1385 10534 20          P=     0
1386 10536 306         LCHEX  6
1387 10539 A02         C=C+A  P             6+15-log2(size)
1388 1053C 486         GOC    PAKROM        Go if remaining ROMs < 1 page
1389 1053F 80D0        P=C    0             Prepare to compute boundary
1390 10543 AC2         C=0    S
1391 10546 A4E         C=C-1  S             =15 if illegal
1392 10549 3312        LCHEX  8421          C[s] has boundary size
           48
1393 1054F 25          P=     5
1394 10551 A96         C=A    WP            Copy table entry nibs
1395 10554 78BE        GOSUB  SIZE10        Compute modsize
1396 10558 ACA         A=C    S             A[s] has boundary size
1397 1055B A3C         A=A-1  X             A[xs] has #pages-1 needed
1398 1055E BCA         C=-C   S
1399 10561 B42  CON130 C=C-A  S             Look at next boundary
1400 10564 493         GOC    CON160        Go if exhausted
1401 10567 80DF        P=C    15            Point at page marker
1402 1056B AA6         C=A    XS            # pages-1 to check
1403 1056E 90D  CON140 ?B#0   P             Page available?
1404 10571 0F          GOYES  CON130        No, check next boundary
1405 10573 0C          P=P+1
1406 10575 A2E         C=C-1  XS            Check adjacent pages?
1407 10578 55F         GONC   CON140        Yes
1408 1057B 170         D1=D1+ 1             No.  found spot!
1409 1057E 812         CSLC
1410 10581 F2          CSL    A
1411 10583 F2          CSL    A
1412 10585 1553        DAT1=C X             Write out address hinibs
1413 10589 1C0         D1=D1- 1
1414 1058C 80D2        P=C    2             Hold hinib to mark map
1415 10590 AA6         C=A    XS            Prepare to mark allocation map
1416 10593 A0D  CON150 B=B-1  P             Mark page as taken
1417 10596 0C          P=P+1
1418 10598 A2E         C=C-1  XS
1419 1059B 57F         GONC   CON150
1420 1059E 179  CON160 D1=D1+ 10
1421 105A1 688F        GOTO   CONF70
1422 105A5 AF2  PAKROM C=0    W             Prepare to compute bubble bdrys
1423 105A8 20          P=     0
```

```
1424 105AA 74BE          GOSUB  FNDBUB          Find lower bubble
1425 105AE 4F0           GOC    PAKR40          Go if one or no bubbles
1426 105B1 BF2           CSL    W
1427 105B4 8E00          GOSUBL =CSLW5          Move first bubble boundary
           00
1428 105BA 74AE          GOSUB  FNDBUB          Find upper bubble
1429 105BE AF5   PAKR40  B=C    W               Hold bubble boundaries
1430 105C1 15F5  PAKR50  C=DAT1 6
1431 105C5 774E          GOSUB  SIZE10          Compute module size
1432 105C9 A38           B=A+B  X               Add to start of first bubble
1433 105CC AF9           C=B    W
1434 105CF 8E00          GOSUBL =CSRW3
           00
1435 105D5 4E0           GOC    PAKR60          Go if overflow first bubble
1436 105D8 9B1           ?B>C   X               Fits in first bubble?
1437 105DB 90            GOYES  PAKR60          No
1438 105DD B3C           A=B-A  X               Yes.  recover bubble address.
1439 105E0 6B30          GOTO   PAKR75          Write out address.
1440 105E4 B38   PAKR60  B=B-A  X               Recover bubble address
1441 105E7 8E00          GOSUBL =CSRW3          Start of second bubble
           00
1442 105ED A3A           A=A+C  X
1443 105F0 453           GOC    PAKR80
1444 105F3 8F00          GOSBVL =CSRC3
           000
1445 105FA 9B6           ?A>C   X               Module fits in second bubble?
1446 105FD 92            GOYES  PAKR80          No
1447 105FF 8F00          GOSBVL =CSLC3
           000
1448 10606 ABE           ACEX   X               Bubble address to a
1449 10609 8E00          GOSUBL =CSLW5
           00
1450 1060F BF2           CSL    W               Align new bubblestart
1451 10612 25            P=     5
1452 10614 A99           C=B    WP
1453 10617 28            P=     8
1454 10619 A95           B=C    WP              Store new bubblestart
1455 1061C 170   PAKR75  D1=D1+ 1
1456 1061F 1513          DAT1=A X
1457 10623 1C0           D1=D1- 1
1458 10626 179   PAKR80  D1=D1+ 10
1459 10629 A6F           D=D-1  B               Any more ROMs?
1460 1062C 549           GONC   PAKR50          Yes
1461 1062F BF7   CON170  DSR    W
1462 10632 BF7           DSR    W               MM i/o device counter
1463 10635 AB0           A=0    X               First addr=20000
1464 10638 A6F   CON180  D=D-1  B               Any more MM i/o devices?
1465 1063B 492           GOC    CON210          No
1466 1063E 1574          C=DAT1 S               Yes, read size nibble
1467 10642 170           D1=D1+ 1
1468 10645 1592          DAT1=A 3               Write out size
1469 10649 178           D1=D1+ 9
1470 1064C AB2           C=0    X
1471 1064F B36           C=C+1  X
1472 10652 B46   CON190  C=C+1  S               Loop to compute size
```

```
1473 10655 480          GOC    CON200
1474 10658 A36          C=C+C  X
1475 1065B 56F          GONC   CON190          B.E.T. (if size reasonable)
1476 1065E A3A   CON200 A=A+C  X              Add to address for next address
1477 10661 66DF         GOTO   CON180          Next device
1478 10665 22    CON210 P=     2
1479               *
1480               * Sort by pdev to perform configuration.
1481               *
1482 10667 79ED         GOSUB  WHLTBL          Sort by port-dev#
1483 1066B 80A          RESET                  Unconfigure all modules
1484 1066E 118          C=R0
1485 10671 135          D1=C                   Point at table again
1486 10674 171          D1=D1+ 2               Point at port#
1487 10677 D1           B=0    A
1488 10679 CD           B=B-1  A
1489 1067B AE1          B=0    B               Rubbish address = FFF00
1490 1067E A6F   CON220 D=D-1  B               Any more entries?
1491 10681 4E6          GOC    CON270          No
1492 10684 15B6         A=DAT1 7               Read relevant table data
1493 10688 20           P=     0
1494 1068A 3210         LCHEX  001
           0
1495 1068F A0C          A=A-1  P               If a=0, loop will result in c=0
1496 10692 A0C   CON225 A=A-1  P               Loop to position port bit
1497 10695 480          GOC    CON227
1498 10698 A36          C=C+C  X               Move port bit
1499 1069B 56F          GONC   CON225
1500 1069E 801   CON227 OUT=C
1501 106A1 25           P=     5
1502 106A3 B04          A=A+1  P               Memory-mapped i/o?
1503 106A6 5B1          GONC   CON240          No
1504 106A9 D6           C=A    A               Yes
1505 106AB AE2          C=0    B
1506 106AE 302          LCHEX  2
1507 106B1 B96          CSR    WP              Configuration address
1508 106B4 787C         GOSUB  WAITKY          Wait for key up.
1509 106B8 805          CONFIG
1510 106BB 179   CON230 D1=D1+ 10
1511 106BE 6FBF         GOTO   CON220
1512 106C2 D2    CON240 C=0    A
1513 106C4 21           P=     1
1514 106C6 3102         LCHEX  20
1515 106CA C6    CON250 C=C+C  A               Loop to compute chip size
1516 106CC B04          A=A+1  P
1517 106CF 5AF          GONC   CON250
1518 106D2 AE0          A=0    B               Clear garbage from address
1519 106D5 DE           ACEX   A               Addr to c, size to a
1520 106D7 26           P=     6
1521 106D9 735C  CON260 GOSUB  WAITKY          Wait for key up.
1522 106DD 805          CONFIG                 Loop to configure chips
1523 106E0 8BD          ?C>=B  A               Is this a rubbish device?
1524 106E3 40           GOYES  CON265          Yes.  do not inc addr.
1525 106E5 C2           C=C+A  A
1526 106E7 A0C   CON265 A=A-1  P               More chips in sequence?
```

```
1527 106EA 5EE          GONC   CON260          Yes
1528 106ED 4DC          GOC    CON230          No.  B.E.T.
1529 106F0 D9    CON270 C=B    A               Retrieve rubbish address
1530 106F2 804          UNCNFG                 Unconfigure rubbish devices
1531 106F5 114          A=R4
1532 106F8 B64          A=A+1  B
1533 106FB 8A8          ?A=0   A               Has r4[a] been disturbed?
1534 106FE 60           GOYES  CON275          No
1535 10700 6A1B         GOTO   CONFRS          Yes.  was interrupted.  restart.
1536 10704 27    CON275 P=     7
1537                 *
1538                 * Sort table by device types.
1539                 * Since address is secondary sort key, IRAMs will fall after
1540                 *   RAMs (conveniently falling into ROM table).
1541                 *
1542 10706 7A4D         GOSUB  WHLTBL          Sort into device types
1543 1070A 7AEC         GOSUB  TBLPTR
1544                 *
1545                 * Sort RAMS by pdev for old/newtable compare
1546                 *
1547 1070E 7B9C         GOSUB  SORTP2          Sort RAMs by port-dev#
1548                 *
1549                 * Sort ROMS by pdev
1550                 *
1551 10712 72EC         GOSUB  TBLPTR
1552 10716 7B1D         GOSUB  ROMTPT
1553 1071A 7F8C         GOSUB  SORTP2          Sort ROMs by port-dev#
1554 1071E 870          ?ST=1  0               Are we doing coldstart?
1555 10721 60           GOYES  coldst          Yes
1556 10723 63A0         GOTO   CON280          No
1557 10727 870   coldst ?ST=1  0               Was coldstart requested?
1558 1072A 90           GOYES  COLDS1          Yes
1559 1072C 8D00         GOVLNG =COLDST         No.  Do a coldstart!
           000
1560 10733 AF0   COLDS1 A=0    W
1561 10736 1F2B         D1=(5) =LOCKWD
           7F2
1562 1073D 1517         DAT1=A W
1563 10741 1E6E         D1=(4) =CONFST         Point at configuration buffers
           9F
1564 10747 1597         DAT1=A 8               Zeroes for cnfend,mainen,bufend
1565 1074B 171          D1=D1+ 2               Point past cnf table
1566 1074E 137          CD1EX
1567 10751 1F85         D1=(5) =MAINST
           5F2
1568                 *
1569                 * Initialize MAINST --> CURREN
1570                 *
1571                 nPTR1  EQU    (((CURREN)-(MAINST))/5)+1
1572 10758 2B           P=     16-(nPTR1)
1573 1075A 7736         GOSUB  INITPT          Initialize mainst,currst,curren
1574 1075E E6           C=C+1  A               Move past 0 byte of program end
1575 10760 E6           C=C+1  A
1576 10762 145          DAT1=C A               Initialize mainen
1577 10765 E6           C=C+1  A               Move past 0 byte of i/o buffers
```

```
1578 10767 E6          C=C+1  A
1579 10769 E6          C=C+1  A
1580 1076B E6          C=C+1  A
1581 1076D 174         D1=D1+ 5
1582            *
1583            * Initialize IOBFEN ---> RFNBFR
1584            *
1585            nPTR2  EQU    (((RFNBFR)-(IOBFEN))/5)+1
1586 10770 2E          P=     16-(nPTR2)     Initialize iobfen-->rfnbfr
1587 10772 7F16        GOSUB  INITPT
1588            *
1589            * Initialize Command Stack
1590            *    Create a doubly linked list of 5 stack items
1591            *
1592 10776 134         DO=C                  C @ RAM position
1593 10779 AF2         C=0    W
1594 1077C 23          P=     3
1595 1077E 303         LC(1)  3              Link list lens = 0 lead, 3 trl
1596 10781 2B          P=     16-5           5 entries
1597 10783 15C5  COLD25 DATO=C 6             Write out links
1598 10787 165         DO=DO+ 6              Move over link
1599 1078A 0C          P=P+1
1600 1078C 56F         GONC   COLD25         Continue til P=0
1601            *
1602            * Initialize RAWBFR ---> AVMEMS
1603            *
1604 1078F 136         CDOEX                 C <-- RAM position
1605             nPTR25 EQU    (((AVMEMS)-(RAWBFR))/5)+1
1606 10792 2B          P=     16-(nPTR25)
1607 10794 7DF5        GOSUB  INITPT         D1 @ RAWBFR
1608            *
1609            * Initialize MTHSTK --> RAMEND
1610            *
1611 10798 1F99        D1=(5) =MTHSTK        AVMEME
           5F2
1612 1079F 119         C=R1
1613            nPTR3  EQU    (((RAMEND)-(MTHSTK))/5)+1
1614 107A2 2A          P=     16-(nPTR3)     Initialize mthstk,forstk,gsbstk,
1615 107A4 7DE5        GOSUB  INITPT         Active,calstk,ramend
1616 107A8 DA          A=C    A
1617 107AA BF0         ASL    W
1618 107AD BF0         ASL    W
1619 107B0 20          P=     0
1620 107B2 31A1        LCHEX  1A
1621 107B6 1596  COLD30 DAT1=A 7             Clear chain list; 27 chains
1622 107BA 176         D1=D1+ 7
1623 107BD A6E         C=C-1  B
1624 107C0 55F         GONC   COLD30
1625 107C3 6383        GOTO   PUTBUF         Store tables in i/o buffers
1626 107C7 7D2C  CON280 GOSUB  TBLPTR
1627 107CB AF1         B=0    W
1628 107CE AE5         B=C    B
1629 107D1 F1          BSL    A
1630 107D3 F1          BSL    A
1631 107D5 F1          BSL    A              #entries in newtable in b[m]
```

```
1632 107D7 1B6E          DO=(5) =CONFST       Start of i/o buffers
           9F2
1633 107DE AF2           C=0    W
1634 107E1 146           C=DAT0 A             Read first header
1635 107E4 B66           C=C+1  B             Old RAMtable?
1636 107E7 460           GOC    CON290        Yes
1637 107EA 6C3F          GOTO   coldst        No.  id doesn't match
1638 107EE 7185  CON290  GOSUB  CDIV10        C/10 to A, B
1639 107F2 103           R3=A                 Hold in R3
1640 107F5 1C9           D1=D1- 10            Will look at lower 9 nibs
1641 107F8 184           DO=DO- 5
1642 107FB 179   CON310  D1=D1+ 10
1643 107FE A5D           B=B-1  M             Anything more in newtable?
1644 10801 405           GOC    CON380        No, remaining old RAMs missing
1645 10804 169   CON320  DO=DO+ 10
1646 10807 A6D           B=B-1  B             Anything more in oldtable?
1647 1080A 435           GOC    CON390        No, remaining newtable RAMs new
1648 1080D 15B8  CON330  A=DAT1 9             Newtable entry
1649 10811 15E8          C=DAT0 9             Oldtable entry
1650 10815 23            P=     3
1651 10817 916           ?A#C   WP            Same size, pdev#, seq pos?
1652 1081A 90            GOYES  CON340        No.
1653 1081C 28            P=     8
1654 1081E 902           ?A=C   P             Chipcount same?
1655 10821 AD            GOYES  CON310        Yes, on to next entries
1656 10823 9B2   CON340  ?A<C   X             Diff devs.  newtable entry new?
1657 10826 B1            GOYES  CON360        Yes.
1658 10828 9B6           ?A>C   X             No.  oldtable entry missing?
1659 1082B E0            GOYES  CON350        Yes.
1660 1082D 7175          GOSUB  MRKNEW        Newtable entry new...mark it
1661 10831 7C75          GOSUB  MRKOLD        Oldtable entry missing...mark it
1662 10835 65CF          GOTO   CON310
1663 10839 7475  CON350  GOSUB  MRKOLD        Oldtable entry missing...mark it
1664 1083D 66CF          GOTO   CON320
1665 10841 7D55  CON360  GOSUB  MRKNEW        Newtable entry is new...mark it
1666 10845 179           D1=D1+ 10
1667 10848 A5D           B=B-1  M             More newtable entries?
1668 1084B 51C           GONC   CON330        Yes
1669 1084E 7F55  CON370  GOSUB  MRKOLD        No.  oldtable dev missing. mark.
1670 10852 169   CON380  DO=DO+ 10
1671 10855 A6D           B=B-1  B             More oldtable entries?
1672 10858 55F           GONC   CON370        Yes
1673 1085B 4F0           GOC    CON400        No.  ready to move memory. B.E.T.
1674 1085E 7045  CON390  GOSUB  MRKNEW        Mark it.
1675 10862 179           D1=D1+ 10
1676 10865 A5D           B=B-1  M
1677 10868 55F           GONC   CON390
1678 1086B 1F49  CON400  D1=(5) =AVMEMS       Prepare to examine missing RAMs
           5F2
1679 10872 143           A=DAT1 A
1680 10875 D8            B=A    A             Hold AVMEMS in b
1681 10877 8F00          GOSBVL =D=AVME       AVMEME in d
           000
1682 1087E 1EDE          D1=(4) (=CONFST)+7   Point at oldtable+2
           9F
```

```
1683 10884 11B          C=R3                    #entries in oldtable
1684 10887 816          CSRC                    Lsn to a[s]
1685 1088A ACA          A=C     S
1686 1088D 816          CSRC                    Msn to c[s]
1687 10890 A4C   CON410 A=A-1   S               Any more entries?
1688 10893 580          GONC    CON420          Yes
1689 10896 A4E          C=C-1   S               Maybe, check msn
1690 10899 474          GOC     CON460          No, done checking
1691 1089C 15F6  CON420 C=DAT1  7               Get oldtable entry nibs 2-8
1692 108A0 25           P=      5
1693 108A2 90A          ?C=0    P               Marked as missing?
1694 108A5 53           GOYES   CON450          No
1695 108A7 80D1         P=C     1               Hold chipsize nibble
1696 108AB AE2          C=0     B               C[a] has addr of missing chip
1697 108AE 8BF          ?C>=D   A               Started after AVMEME?
1698 108B1 52           GOYES   COLD            Yes, better coldstart
1699 108B3 8B1          ?C<B    A               Started before AVMEMS?
1700 108B6 02           GOYES   COLD            Yes, better coldstart
1701 108B8 D0           A=0     A
1702 108BA B24          A=A+1   XS
1703 108BD B24          A=A+1   XS
1704 108C0 C4    CON430 A=A+A   A               Loop to compute chipsize
1705 108C2 0C           P=P+1
1706 108C4 5BF          GONC    CON430
1707 108C7 26           P=      6
1708 108C9 C2    CON440 C=C+A   A               Loop to find endaddr
1709 108CB A0E          C=C-1   P
1710 108CE 5AF          GONC    CON440
1711 108D1 8BB          ?C<=D   A               RAM ends after AVMEME?
1712 108D4 60           GOYES   CON450          No.  was entirely empty.
1713 108D6 605E  COLD   GOTO    coldst
1714 108DA 179   CON450 D1=D1+  10              On to next entry
1715 108DD 62BF         GOTO    CON410
1716 108E1 74AB  CON460 GOSUB   C=RAME          Current end of RAM
1717 108E5 E3           D=D-C   A               Old AVMEME-old RAMEND
1718 108E7 119          C=R1                    New RAMEND
1719 108EA CB           C=C+D   A
1720 108EC 10B          R3=C    A               New AVMEME
1721 108EF 750B         GOSUB   TBLPTR
1722 108F3 26           P=      6
1723            *
1724            * Sort RAMs by address to restore memory integrity.
1725            *
1726 108F5 76BA         GOSUB   SORT            Sort newtable by address
1727 108F9 7BFA         GOSUB   TBLPTR
1728 108FD D5           B=C     A               Entry counter to b
1729 108FF 723B         GOSUB   ROMTPT          Point to end of RAMtable
1730 10903 119          C=R1                    RAMEND
1731 10906 134          D0=C                    Lowbound of bubble
1732 10909 12B          CR3EX                   Highbound of bubble
1733 1090C D7           D=C     A               New AVMEME
1734 1090E 841          ST=0    1
1735 10911 1C9   CON470 D1=D1-  10
1736 10914 A6D          B=B-1   B               Any more table entries?
1737 10917 531          GONC    CON480          Yes
```

```
1738 1091A 851          ST=1   1          No more plugins.  prepare to move
1739 1091D 23           P=     3          Data up from internal RAM.
1740 1091F 35F0         LCHEX  20000F     3 chips, 4 kbits each
           0002
1741 10927 6220         GOTO   CON490     Do it
1742 1092B 15F8 CON480  C=DAT1 9          Read next table entry
1743 1092F 27           P=     7
1744 10931 90A          ?C=0   P          Was it marked as new?
1745 10934 61           GOYES  CON490     No, will move bubble
1746 10936 7DCA         GOSUB  MSIZE      Yes, module size to a
1747 1093A F0           ASL    A
1748 1093C F0           ASL    A
1749 1093E 136          CDOEX
1750 10941 E2           C=C-A  A
1751 10943 136          CDOEX             Move lowbound to expand bubble
1752 10946 6ACF         GOTO   CON470
1753 1094A 79BA CON490  GOSUB  MSIZE      Prepare to move bubble
1754 1094E F0           ASL    A
1755 10950 F0           ASL    A
1756 10952 11B          C=R3              Highbound of bubble
1757 10955 132          ADOEX             Lowbound of bubble
1758 10958 8A6          ?A#C   A          Bubblesize=0?
1759 1095B 51           GOYES  CON500     No, must move data
1760 1095D 132          ADOEX             Yes, just change pointers
1761 10960 E2           C=C-A  A          New highbound&lowbound
1762 10962 10B          R3=C
1763 10965 134          DO=C
1764 10968 861          ?ST=0  1          Was this disp driver RAM?
1765 1096B 6A           GOYES  CON470     No...on to next module
1766 1096D 5C7          GONC   CON550     Yes...done!  B.E.T.
1767 10970 132  CON500  ADOEX             Get back size
1768 10973 E2           C=C-A  A          New highbound
1769 10975 8BF          ?C>=D  A          Below AVMEME?
1770 10978 40           GOYES  CON510     No
1771 1097A DB           C=D    A          Yes, use AVMEME
1772 1097C 113  CON510  A=R3              Old highbound
1773 1097F EA           A=A-C  A          #nibs to move
1774 10981 486          GOC    CON550     If negative, we are done
1775 10984 8A8          ?A=0   A
1776 10987 36           GOYES  CON550     If zero, we are done
1777 10989 11B          C=R3              Old highbound
1778 1098C 137          CD1EX
1779 1098F 10B          R3=C              Save RAMtable pointer
1780 10992 D6           C=A    A          #nibs to move
1781 10994 8E00         GOSUBL =MOved3    Move down (to higher address)
           00
1782 1099A 11B  CON540  C=R3              Recover RAMtable pointer
1783 1099D 137          CD1EX
1784 109A0 10B          R3=C              New highbound
1785 109A3 871          ?ST=1  1          Was this disp driver RAM?
1786 109A6 44           GOYES  CON550     Yes... done!
1787 109A8 686F         GOTO   CON470
```

```
1788                    STITLE CNFFND Utility
1789                ****************************************************************
1790                ****************************************************************
1791                **
1792                ** Name:(S) CNFFND  -  Configuration Buffer Find
1793                **
1794                ** Category:   CONFIG
1795                **
1796                ** Purpose:    FINDS CONFIGURATION BUFFER
1797                **
1798                ** Entry:    C(B) IS BUF ID#
1799                **
1800                ** Exit:   C(B)= BUFFER ID# (preserved from input)
1801                **           CARRY SET=> MATCH FOUND
1802                **                       D1   POINTS PAST BUFFER HEADER
1803                **                       A(A) BUFFER LENGTH
1804                **                       SB=0
1805                **           CARRY CLR=> NO MATCH
1806                **
1807                ** Calls:    none
1808                **
1809                ** Stack lvls: 0
1810                **
1811                ** Uses:     A(A), D1
1812                **
1813                ** Detail:   Length given in header reflects the amount of
1814                **           scratch area in the buffer, but doesn't include
1815                **           the total buffer area (e.g. the 5 nibbles used
1816                **           by the header)
1817                **
1818                ** History:
1819                **
1820                **    Date      Programmer        Modifications
1821                **  --------    ----------        -------------
1822                **  07/04/82    SW                Added documentation
1823                **  02/11/83    NM                Moved to CNF module
1824                **
1825                ****************************************************************
1826                ****************************************************************
1827                *
1828 109AC 1F6E  =CNFFND D1=(5) =CONFST        HARD-WIRED START OF CONF AREA
          9F2
1829 109B3 143    CNFND3 A=DAT1 A
1830 109B6 174           D1=D1+ 5             POINT PAST HEADER
1831 109B9 96C           ?A#0   B             NOT AT END OF BUFFER AREA?
1832 109BC 40            GOYES  CNFND5
1833 109BE 01            RTN
1834                *
1835 109C0 B6A    CNFND5 A=A-C  B
1836 109C3 822           SB=0
1837 109C6 F4            ASR    A
1838 109C8 F4            ASR    A
1839                * SB=0 => FOUND BUF# MATCH (E.G. SHIFTED OFF ZEROES)
1840 109CA 832           ?SB=0
1841 109CD 00            RTNYES
```

```
1842                *
1843 109CF 137            CD1EX
1844 109D2 C2             C=C+A  A              POINT TO NEXT BUFFER
1845 109D4 137            CD1EX
1846 109D7 5BD            GONC   CNFND3         (B.E.T.)
1847                *
1848 109DA 23     RST<R3  P=     3
1849 109DC 8C00           GOLONG =Rstk<R
           00
1850 109E2 23     R3<RST  P=     3
1851 109E4 8C00           GOLONG =R<Rstk
           00
```

```
1852                    STITLE Configuration code (cont.)
1853 109EA 7A0A  CON550 GOSUB  TBLPTR          Prepare to bubble up
1854 109EE D5           B=C    A               Entry counter to b
1855 109F0 1B49         D0=(5) =AVMEMS
             5F2
1856 109F7 146          C=DAT0 A
1857 109FA D7           D=C    A               AVMEMS to d
1858 109FC D2           C=0    A
1859 109FE 24           P=     4
1860 10A00 303          LCHEX  3               C contains 30000
1861 10A03 10B          R3=C                   Lowbound of bubble
1862 10A06 134          D0=C                   Highbound of bubble
1863 10A09 A6D   CON560 B=B-1  B               Any more modules?
1864 10A0C 475          GOC    CON610          No
1865 10A0F 15F8         C=DAT1 9               Read next entry
1866 10A13 27           P=     7
1867 10A15 90A          ?C=0   P               Was it marked as new?
1868 10A18 91           GOYES  CON580          No, will move bubble
1869 10A1A 79E9         GOSUB  MSIZE           Yes, will grow bubble by size
1870 10A1E F0           ASL    A
1871 10A20 F0           ASL    A
1872 10A22 136          CDOEX
1873 10A25 C2           C=C+A  A               New upper bound
1874 10A27 136          CDOEX
1875 10A2A 179   CON570 D1=D1+ 10
1876 10A2D 6BDF         GOTO   CON560          On to next entry
1877 10A31 72D9  CON580 GOSUB  MSIZE           Prepare to move bubble
1878 10A35 F0           ASL    A
1879 10A37 F0           ASL    A
1880 10A39 11B          C=R3                   Lowbound of bubble
1881 10A3C 132          ADOEX                  Highbound of bubble
1882 10A3F 8A6          ?A#C   A               Bubblesize=0?
1883 10A42 11           GOYES  CON590          No, will move memory
1884 10A44 132          ADOEX                  Yes, change pointers only
1885 10A47 C2           C=C+A  A
1886 10A49 10B          R3=C                   New lowbound
1887 10A4C 134          D0=C                   New highbound
1888 10A4F 6ADF         GOTO   CON570
1889 10A53 132   CON590 ADOEX                  Recover size
1890 10A56 C2           C=C+A  A               New lowbound
1891 10A58 8BB          ?C<=D  A               Above AVMEMS?
1892 10A5B 40           GOYES  CON600          No
1893 10A5D DB           C=D    A               Yes, use AVMEMS
1894 10A5F 113   CON600 A=R3                   Old lowbound
1895 10A62 E2           C=C-A  A               # nibs to move
1896 10A64 432   CON610 GOC    CON650          Done if negative
1897 10A67 8AA          ?C=0   A
1898 10A6A E1           GOYES  CON650          Done if zero
1899 10A6C 12B          CR3EX                  Old lowbound
1900 10A6F 137          CD1EX
1901 10A72 12B          CR3EX                  Save RAMtable ptr in r3
1902 10A75 8E00         GOSUBL =Moveu3         Move up (to lower address)
             00
1903 10A7B 11B          C=R3
1904 10A7E 137          CD1EX                  Restore RAMtable pointer
```

```
1905 10A81 10B          R3=C                    Restore lowbound
1906 10A84 65AF         GOTO    CON570
1907 10A88 7C69  CON650 GOSUB   TBLPTR          RAMtable pointer
1908 10A8C 1C2          D1=D1-  3               Point at device type nibble
1909 10A8F AA0          A=0     XS
1910 10A92 179   CON660 D1=D1+  10              Loop to unmark new RAMs
1911 10A95 A6E          C=C-1   B               Any more entries?
1912 10A98 490          GOC     CON670          No
1913 10A9B 1512         DAT1=A  XS              Yes, write out zero
1914 10A9F 52F          GONC    CON660          B.E.T.
1915 10AA2 73E9  CON670 GOSUB   C=RAME          Old RAMEND
1916 10AA6 111          A=R1                    New RAMEND
1917 10AA9 EA           A=A-C   A               Old-new
1918 10AAB 2A           P=      16-(nPTR3)
1919 10AAD C2    CON680 C=C+A   A               Update RAMEND,clcvar,calstk,
1920 10AAF 145          DAT1=C  A               Active,gsbstk,AVMEME
1921 10AB2 1C4          D1=D1-  5
1922 10AB5 147          C=DAT1  A
1923 10AB8 0C           P=P+1
1924 10ABA 52F          GONC    CON680
1925 10ABD 1E9B         D1=(4)  (=PRMPTR)+2     Prepare to update chain ptrs
          5F
1926 10AC3 31A1         LC(2)   26              There are 27 chain ptrs
1927 10AC7 D5           B=C     A               Counter
1928 10AC9 147   CON685 C=DAT1  A
1929 10ACC C2           C=C+A   A
1930 10ACE 145          DAT1=C  A
1931 10AD1 176          D1=D1+  7
1932 10AD4 A6D          B=B-1   B
1933 10AD7 51F          GONC    CON685
1934 10ADA 7A19         GOSUB   TBLPTR
1935              *
1936              * Sort RAM by pdev for old/newtable compare next time.
1937              *
1938 10ADE 7BC8         GOSUB   SORTP2          Sort RAMs by port-dev#
1939              *
1940              * Determine if system configuration has changed.
1941              *
1942 10AE2 7219         GOSUB   TBLPTR
1943 10AE6 7B49         GOSUB   ROMTPT          D1 points at ROMTBL, D[B]=#ents
1944 10AEA 1B8E         DO=(5)  (=CONFST)+2     Point at size or RAM oldtable
          9F2
1945 10AF1 D2           C=0     A
1946 10AF3 1563         C=DATO  X
1947 10AF7 162          DO=DO+  3               Point past size field.
1948 10AFA 132          ADOEX
1949 10AFD CA           A=A+C   A
1950 10AFF 132          ADOEX                   Point past RAM oldtable.
1951 10B02 146          C=DATO  A               Start of ROMtable conftable
1952 10B05 B66          C=C+1   B
1953 10B08 B66          C=C+1   B               Is this expected ID?
1954 10B0B 460          GOC     ROMC10          Yes.
1955 10B0E 681C         GOTO    coldst          No.
1956 10B12 164   ROMC10 DO=DO+  5               DO -> oldtable, A[A]=size*100H
1957 10B15 7A52         GOSUB   CDIV10          #entries to B[B]
```

```
1958 10B19 29          P=     9            To compare table entries
1959 10B1B A6D  ROMC20 B=B-1  B            Any more in oldtable?
1960 10B1E 482         GOC    ROMC50       No.  Everything in order.
1961 10B21 1521        A=DAT0 WP           Yes.  Read oldtable entry.
1962 10B25 169         DO=DO+ 10           And point past it.
1963 10B28 A6F  ROMC30 D=D-1  B            Any more in newtable?
1964 10B2B 481         GOC    ROMC40       No.  We've found a missing ROM.
1965 10B2E 1571        C=DAT1 WP           Yes.  Read newtable entry.
1966 10B32 179         D1=D1+ 10           And point past it.
1967 10B35 9B6         ?C<A   X            New entry before old?
1968 10B38 0F          GOYES  ROMC30       Yes.  Get next new entry.
1969 10B3A 9B2         ?C>A   X            New entry after old?
1970 10B3D 70          GOYES  ROMC40       Yes.  We've found a missing ROM.
1971 10B3F 912         ?C=A   WP           Entry identical, incl. address?
1972 10B42 9D          GOYES  ROMC20       Yes.  On to next oldtable entry.
1973 10B44 853  ROMC40 ST=1   3            Mark ROM configuration changed.
1974 10B47      ROMC50                      Here if ROM config unchanged.
1975 10B47 7DA8 PUTBUF GOSUB  TBLPTR
1976 10B4B 76E8        GOSUB  ROMTPT
1977 10B4F 72E8        GOSUB  ROMTPT       Point at MMI/O table
1978           *
1979           * Sort MMIO by pdev to establish an order.
1980           *
1981 10B53 7658        GOSUB  SORTP2       Sort table by port-dev#
1982 10B57 7D98        GOSUB  TBLPTR
1983 10B5B 76D8        GOSUB  ROMTPT       Get past RAM table
1984 10B5F 133         AD1EX
1985 10B62 D8          B=A    A            Hold ROMtable pointer
1986 10B64 131         D1=A
1987 10B67 7AC8        GOSUB  ROMTPT       Get past ROM table
1988 10B6B 133         AD1EX
1989 10B6E 103         R3=A                Hold MMIO pointer
1990 10B71 131         D1=A
1991 10B74 7DB8        GOSUB  ROMTPT       Get past MMIO table
1992 10B78 137         CD1EX
1993 10B7B DD          BCEX   A            End ptr to b
1994 10B7D 135         D1=C                ROM table pointer
1995 10B80 841         ST=0   1
1996 10B83 AF3         D=0    W            #deleted entries counter
1997 10B86 173  CON690 D1=D1+ 4            Point at addr nibs
1998 10B89 D2          C=0    A
1999 10B8B CE          C=C-1  A            Rubbish address = FFF
2000 10B8D 133  CON700 AD1EX               ROMtable pointer
2001 10B90 12B         CR3EX               MMIO table pointer
2002 10B93 8B6         ?A>C   A            Past ROM table?
2003 10B96 95          GOYES  CON720       Yes
2004 10B98 12B         CR3EX               No
2005 10B9B 133         AD1EX
2006 10B9E 1533        A=DAT1 X            Read address
2007 10BA2 932         ?A=C   X            Rubbish ROM?
2008 10BA5 90          GOYES  CON710       Yes
2009 10BA7 179         D1=D1+ 10           No.  next ROM.
2010 10BAA 62EF        GOTO   CON700
2011 10BAE 1C3  CON710 D1=D1- 4            Begin dest
2012 10BB1 137         CD1EX
```

```
2013 10BB4 135          D1=C
2014 10BB7 134          DO=C
2015 10BBA 10A          R2=C                Hold begin dest
2016 10BBD 169          DO=DO+ 10           Begin src
2017 10BC0 136          CDOEX
2018 10BC3 134          DO=C
2019 10BC6 E9           C=C-B  A            -length
2020 10BC8 FA           C=-C   A            Length
2021 10BCA 8E00         GOSUBL =Moveu3      Pack out entry
           00
2022 10BD0 11A          C=R2
2023 10BD3 135          D1=C                Restore begin dest
2024 10BD6 D2           C=0    A
2025 10BD8 30A          LCHEX  A
2026 10BDB E1           B=B-C  A            Update end pointer
2027 10BDD 123          AR3EX
2028 10BE0 EA           A=A-C  A            Update MMIO table pointer
2029 10BE2 123          AR3EX
2030 10BE5 851          ST=1   1            Indicate config error
2031 10BE8 B67          D=D+1  B            Update deleted entry ctr
2032 10BEB 6A9F         GOTO   CON690
2033 10BEF 29   CON720  P=     9
2034 10BF1 7650         GOSUB  DSLW-P       Align with ROM entry count
2035 10BF5 118          C=RO
2036 10BF8 B7B          C=C-D  W            Update ROM entry count
2037 10BFB 2C           P=     12
2038 10BFD 7A40         GOSUB  DSLW-P       Align with sum
2039 10C01 B7B          C=C-D  W            Update sum
2040 10C04 10B          R3=C                Save counters & tbl pointer
2041 10C07 7A88         GOSUB  csrw6
2042 10C0B BF6          CSR    W
2043 10C0E BF6          CSR    W
2044 10C11 F6           CSR    A
2045 10C13 F6           CSR    A
2046 10C15 F6           CSR    A            Sum of # entries
2047 10C17 C6           C=C+C  A
2048 10C19 D5           B=C    A
2049 10C1B C6           C=C+C  A
2050 10C1D C6           C=C+C  A
2051 10C1F C1           B=B+C  A            10=#entries
2052 10C21 347F         LC(5)  (=CONFST)+17
           9F2
2053 10C28 C1           B=B+C  A            Space for tables + CONFST
2054 10C2A 8E00         GOSUBL =C=MAIN      Using D1, read MAINST into C(A)
           00
2055 10C30 E1           B=B-C  A            Amt to move memory for new tables
2056 10C32 522          GONC   CON730       Go if tables grew
2057 10C35 DA           A=C    A            Begin source
2058 10C37 C9           C=B+C  A            Begin dest
2059 10C39 135          D1=C
2060 10C3C 8F00         GOSBVL =MOVEUA      Move memory to lower address
           000
2061 10C43 6080         GOTO   CON770       Go and adjust references
2062 10C47 6FDA CONCLD  GOTO   coldst
2063 10C4B BF3  DSLW-P  DSL    W
```

```
2064 10C4E 0C           P=P+1
2065 10C50 5AF          GONC   DSLW-P
2066 10C53 03           RTNCC
2067 10C55 1F99 CON730  D1=(5) =AVMEME
           5F2
2068 10C5C 143          A=DAT1 A              (AVMEME)
2069 10C5F 1C4          D1=D1- (AVMEME)-(AVMEMS)
2070 10C62 147          C=DAT1 A              (AVMEMS)
2071 10C65 EA           A=A-C  A              Available memory
2072 10C67 4FD          GOC    CONCLD         BAD NEWS...COLDSTART
2073 10C6A D2           C=0    A
2074 10C6C 314D         LC(2)  =LEEWAY
2075 10C70 EA           A=A-C  A              Available memory - leeway
2076 10C72 540          GONC   CON740         Still non-negative
2077 10C75 D0           A=0    A              Tight!  Keep tables same size.
2078 10C77 E0  CON740   A=A-B  A              New table shrinks -(this much)
2079 10C79 532          GONC   CON760         Room for the whole thing
2080 10C7C D3           D=0    A              For counter.
2081 10C7E 31A0         LC(2)  10             For shrink loop.
2082 10C82 851          ST=1   1              Indicate CONFIG error.
2083 10C85 E7  CON750   D=D+1  A              Increment excluded entries ctr.
2084 10C87 E1           B=B-C  A              Reduce expand amount.
2085 10C89 CA           A=A+C  A              Now do we fit?
2086 10C8B 59F          GONC   CON750         No.
2087 10C8E 25           P=     5
2088 10C90 77BF         GOSUB  DSLW-P         Align excluded ctr w/total ctr
2089 10C94 11B          C=R3
2090 10C97 B7B          C=C-D  W
2091 10C9A 10B          R3=C                  New counters.
2092 10C9D 8A9  CON760  ?B=0   A              How much do tables grow?
2093 10CA0 B3           GOYES  CON775
2094 10CA2 1B49         D0=(5) =AVMEMS
           5F2
2095 10CA9 142          A=DAT0 A
2096 10CAC 131          D1=A                  End Source
2097 10CAF C0           A=A+B  A              End Dest
2098 10CB1 133          AD1EX
2099 10CB4 1A85         D0=(4) =MAINST
           5F
2100 10CBA 146          C=DAT0 A              Start Source
2101 10CBD 8F00         GOSBVL =MOVED2        Move memory to higher address.
           000
2102 10CC4 1B85 CON770  D0=(5) =MAINST
           5F2
2103 10CCB 146          C=DAT0 A
2104 10CCE DA           A=C    A              For RFADJ
2105 10CD0 C9           C=C+B  A              Update MAINST
2106 10CD2 144          DAT0=C A
2107 10CD5 8E00         GOSUBL =RFADJ+        Adjust references.
           00
2108 10CDB 11B  CON775  C=R3
2109 10CDE 8E71         GOSUBL TBLPT+         D1->oldtable; ctrs in D
           7F
2110 10CE4 AFB          C=D    W
2111 10CE7 8E8A         GOSUBL csrw6          C[B]=total of all tables
```

```
                7F
2112 10CED D5           B=C     A
2113 10CEF 1B6E         D0=(5)  =CONFST
           9F2
2114 10CF6 A4E   CON780 C=C-1   S               For buffer ID
2115 10CF9 80DF         P=C     15
2116 10CFD 89C          ?P=     12              About to write table FC?
2117 10D00 94           GOYES   CON810          Yes.  Done.
2118 10D02 D2           C=0     A
2119 10D04 AEB          C=D     B               #entries in this buffer
2120 10D07 BF7          DSR     W
2121 10D0A F7           DSR     A
2122 10D0C B61          B=B-C   B               Subtract from total
2123 10D0F 570          GONC    CON790          Go if room for whole table
2124 10D12 A69          C=B+C   B               Only write this many entries
2125 10D15 D1           B=0     A               No more room
2126 10D17 DA    CON790 A=C     A
2127 10D19 C6           C=C+C   A
2128 10D1B C6           C=C+C   A
2129 10D1D C2           C=A+C   A
2130 10D1F C6           C=C+C   A               Size of table
2131 10D21 F2           CSL     A
2132 10D23 F2           CSL     A
2133 10D25 A6E          C=C-1   B               C[B]=FF
2134 10D28 80F0         CPEX    0               C[B]=Buffer ID
2135 10D2C 144          DAT0=C  A               Write out buffer ID.
2136 10D2F 164          D0=D0+  5
2137 10D32 CC    CON800 A=A-1   A               Any more entries to copy?
2138 10D34 41C          GOC     CON780          No.  On to next table.
2139 10D37 15F9         C=DAT1  10              Read next entry.
2140 10D3B 15C9         DAT0=C  10              Write to CNF table.
2141 10D3F 169          D0=D0+  10
2142 10D42 179          D1=D1+  10
2143 10D45 6CEF         GOTO    CON800          Back for more.
2144 10D49 D2    CON810 C=0     A
2145 10D4B 14C          DAT0=C  B               Zero byte at end.
2146 10D4E 8F00         GOSBVL  =CLRXDS
           000
2147 10D55 09           C=ST
2148 10D57 816          CSRC
2149 10D5A 10C          R4=C                    Hold status bits
2150 10D5D 8080         INTON
2151 10D61 8E00         GOSUBL  =CLKSPD         Compute clkspd, store -> CSPEED
           00
2152 10D67 7F6C         GOSUB   RST<R3          Restore 3 subroutine levels
2153 10D6B 7850         GOSUB   STMBF?          Create stmt buffer if necessary
2154 10D6F 6570         GOTO    LXBF++          Create lex files buffer
```

```
2155                    $TITLE More CONFIG utilities
2156                ********************************************************************
2157                ********************************************************************
2158                **
2159                ** Name:    CDIV10  -  A = C/10
2160                **
2161                ** Category:   LOCAL
2162                **
2163                ** Purpose:
2164                **      Divide a 3-digit hex number by 10.
2165                **
2166                ** Entry:
2167                **      Argument in C[2-4].
2168                **
2169                ** Exit:
2170                **      Result in A[B],B[B].
2171                **
2172                ** Calls:      None.
2173                **
2174                ** Uses.......
2175                **              A,B,C
2176                **
2177                ** Stk lvls:   0
2178                **
2179                ** Algorithm:
2180                **      Based on identity 1/5=1/4-1/16+1/64-1/256...
2181                **
2182                ** History:
2183                **
2184                **    Date      Programmer              Modification
2185                **  --------   ----------   ----------------------------------
2186                **  09/14/82   NM           Pulled out of config code
2187                **
2188                ********************************************************************
2189                ********************************************************************
2190 10D73 F6    CDIV10 CSR    A
2191 10D75 F6           CSR    A               Arg to c[x]
2192 10D77 AF0          A=0    W
2193 10D7A 26           P=     6
2194 10D7C A7A   CDIV20 A=A+C  W               Loop to divide by 5 based on
2195 10D7F 81C          ASRB                   Identity:
2196 10D82 81C          ASRB                   1/5=1/4-1/16+1/64-1/256...
2197 10D85 BFA          C=-C   W               Only need 6 iterations here
2198 10D88 0D           P=P-1
2199 10D8A 51F          GONC   CDIV20
2200 10D8D 81C          ASRB                   /2=size/10
2201 10D90 AE8          B=A    B
2202 10D93 01           RTN
2203                ********************************************************************
2204                ********************************************************************
2205                **
2206                ** Name:    INITPT  -  Loop To Initialize Pointers
2207                **
2208                ** Category:   ADDCAL
2209                **
```

```
2210            ** Purpose:
2211            **      Initialize pointers.
2212            **
2213            ** Entry:
2214            **      C[A] = value to initialize with.
2215            **      D1 points at first pointer.
2216            **      P = 16 - (#pointers to initialize).
2217            **
2218            ** Exit:
2219            **      D1 points past last pointer initialized.
2220            **      P=0.
2221            **      Carry clear.
2222            **
2223            ** Calls:      None.
2224            **
2225            ** Uses.......
2226            **  Exclusive: D1,P.
2227            **
2228            ** Stk lvls:   0
2229            **
2230            ** Detail:
2231            **      Used to initialize bunches of pointers at coldstart.
2232            **
2233            ** History:
2234            **
2235            **    Date      Programmer              Modification
2236            **  --------   ----------   ----------------------------------
2237            **  06/21/82   NM           Added documentation
2238            **
2239            ********************************************************************
2240            ********************************************************************
2241 10D95 145  =INITPT DAT1=C A
2242 10D98 174          D1=D1+ 5
2243 10D9B 0C           P=P+1
2244 10D9D 57F          GONC   INITPT
2245 10DA0 03           RTNCC
2246            ********************************************************************
2247            ********************************************************************
2248            **
2249            ** Name:    MRKNEW  -  Mark New Table Entries
2250            **
2251            ** Category:   LOCAL
2252            **
2253            ** Purpose:
2254            **      Mark newtable ID to identify RAM as new (not existing
2255            **      in old configuration table).
2256            **
2257            ** Entry:
2258            **      D1 points at table entry.
2259            **
2260            ** Exit:
2261            **      Modified lower 8 nibs in A and written out at D1.
2262            **      P=7.
2263            **
2264            ** Calls:      None.
```

```
2265                **
2266                ** Uses.......
2267                **  Exclusive: A[7-0],P.
2268                **
2269                ** Stk lvls:   0
2270                **
2271                ** Detail:
2272                **      Changes nibble 7 of table entry from 0 to 1.
2273                **
2274                ** History:
2275                **
2276                **    Date      Programmer              Modification
2277                **  --------   ----------   ----------------------------------
2278                **  06/21/82   NM           Added documentation
2279                **
2280                ********************************************************************
2281                ********************************************************************
2282 10DA2 15B7   MRKNEW A=DAT1 8
2283 10DA6 27             P=     7
2284 10DA8 B04            A=A+1  P              Modify nibble 7
2285 10DAB 1597           DAT1=A 8              Write out
2286 10DAF 01             RTN
2287                ********************************************************************
2288                ********************************************************************
2289                **
2290                ** Name:    MRKOLD  -  Mark Oldtable To Identify Missing RAM
2291                **
2292                ** Category:   LOCAL
2293                **
2294                ** Purpose:
2295                **      Mark oldtable entry to identify RAM as missing (in
2296                **      oldtable but not in newtable).
2297                **
2298                ** Entry:
2299                **      D0 points at table entry.
2300                **
2301                ** Exit:
2302                **      Possibly modified table entry nibs 0-7 in C.
2303                **
2304                ** Calls:      None.
2305                **
2306                ** Uses.......
2307                **  Exclusive: C[7-0],D0,P
2308                **
2309                ** Stk lvls:   0
2310                **
2311                ** Detail:
2312                **      If RAM is missing and was not intentionally removed
2313                **      (size nib set to 0 by FREEPORT means intentional
2314                **      removal), then sets nibble 7 to 1.
2315                **
2316                ** History:
2317                **
2318                **    Date      Programmer              Modification
2319                **  --------   ----------   ----------------------------------
```

```
2320            **  06/21/82   NM          Added documentation
2321            **
2322            ********************************************************************
2323            ********************************************************************
2324 10DB1 15E7  MRKOLD C=DAT0 8
2325 10DB5 23           P=     3
2326 10DB7 90A          ?C=0   P             RAM intentionally removed?
2327 10DBA 00           RTNYES
2328 10DBC 27           P=     7
2329 10DBE 301          LCHEX  1             Modify nibble 7
2330 10DC1 15C7         DAT0=C 8             Write out
2331 10DC5 01           RTN
```

```
2332                    STITLE LEXBUF - Set up Lang Ext Table
2333                ****************************************************************
2334                ****************************************************************
2335                **
2336                ** Name:    LEXBUF  -  Set Up LEX Files Buffer
2337                ** Name:(S) LEXBF+  -  Set Up LEX Files Buffer
2338                **
2339                ** Category:   CONFIG
2340                **
2341                ** Purpose:
2342                **      Set up Language Extension Files Table Buffer
2343                **      Must be called whenever Configuration or # Lex Files
2344                **      changes
2345                **
2346                ** Entry:
2347                **      LEXBUF:  At power on (through CONF)
2348                **               If coldstart
2349                **                  Statement Buffer created
2350                **
2351                **      LEXBF+:  When Lex file copied into RAM
2352                **               Statement Buffer not created
2353                **
2354                ** Exit:
2355                **      Return after Fast POLL for Configuration
2356                **
2357                **      If not enough memory to add all Lex files to Buffer
2358                **         Lex Buffer is collapsed down
2359                **         XROM01 and MAINT are added to Lex Buffer
2360                **
2361                ** Calls:      I/OAL+,LEXFOO,LEXFND,ROMCHK,ROMFND,POLL,I/OCOL
2362                **             R<RSTK,RSTK<R
2363                **
2364                ** Uses:
2365                **  Exclusive: A,B,C,D1
2366                **             RSTKBF (3 levels)
2367                **               Needed for pCONF can be issued
2368                **  Inclusive: A,B,C,D,D0,D1,R1,R2,R3
2369                **
2370                **             R1 = Pointer to next entry in ROM Config Table
2371                **                = Length remaining in ROM Config Table
2372                **
2373                ** Stk lvls:   4
2374                **            +4 levels saved in RSTKBF
2375                **             Allows LEXFND to use 4 lvls, also
2376                **
2377                ** NOTE:
2378                **      The Statement Buffer must be created FIRST in the
2379                **      I/O Buffer area.  Since the LEX Buffer size can
2380                **      change between Power ON and the Statement Buffer
2381                **      may be in use, updating PCADDR that points into
2382                **      the Statement Buffer would be near IMPOSSIBLE, since
2383                **      an offset is not easy to calculate.
2384                **
2385                ** Algorithm:
2386                **
```

```
2387          ** LEXBUF: If Coldstart    (S0=1)
2388          **            Create Statement Buffer         (I/OAL+)
2389          ** LEXBF+  Allocate Language Extension Buffer (I/OAL+)
2390          **            ID=FC, Size=0
2391          **          Save 4 stack levels               (R<RSTK)
2392          **          Search for LEX files in RAM       (LEXFND)
2393          **          Check if ROM Table is non-empty   (ROMCHK)
2394          **          If ROM Config Table NOT empty
2395          **             Search ROM for LEX files & Update LEX Table
2396          **             Repeat until (End of ROM Table)
2397          **               Find next ROM  (ROMFND)
2398          **               Search ROM & Update LEX Table (LEXFND)
2399          **               If not enough memory to Expand (Carry Clear)
2400          **       1:         Collaspe Lex Buffer       (I/OCOL)
2401          **                  Set C(S) so I/OEX1 will not use Leeway
2402          **                  goto 2;
2403          **          Set C(S) so I/OEX1 will use Leeway
2404          **       2: Add Built-in XROM, MAINT to LEX Table Buffer
2405          **             Set R3 @ "00" byte to indicate end of file
2406          **             Set D0 @ start of XROM01
2407          **             Add xrom01 and MAINT to LEX Buffer (LEXF00)
2408          **             If not enough memory to add --> goto 1;
2409          **          CONFIGURATION Poll.
2410          **          Restore return levels to stack    (RSTK<R)
2411          **          If handled, restart CONFIGURATION from the
2412          **            beginning.
2413          **          else
2414          **            go Auto delete I/O buffers
2415          **
2416          ** Detail:
2417          **       xrom01 and MAINT Lex Files are CHAINED together.
2418          **       The next Lex File relative address pointer within
2419          **       xrom01 points to the start of MAINT.  One call to
2420          **       LEXF00 will add both xrom01 and MAINT to the Lex
2421          **       Buffer.
2422          **
2423          **       4 stack levels are saved to fixed TWO problems:
2424          **
2425          **       Within LEXFND (called by LEXBUF)
2426          **         Usage is 4 levels  (Stack save, I/OEX2 (uses2)
2427          **         One level too deep ---
2428          **
2429          **       pCONF issued at end of LEXBUF
2430          **         Since FPOLL uses 2 levels to get there
2431          **         No levels left for HPIL/Lex file to deal with
2432          **            its buffers
2433          **
2434          **       COPY
2435          **         COPYu
2436          **           RSTK <-- R1
2437          **             LEXBUF
2438          **
2439          ** History:
2440          **
2441          **    Date     Programmer             Modification
```

```
2442            **  --------   ----------   ----------------------------------
2443            **  07/09/82   JP           Modified documentation
2444            **  09/09/82   JP           Add no memory to expand handling
2445            **  11/01/82   JP           Added New Lex file format
2446            **  11/04/82   JP           Calling LEXF00 to add xrom01/MAINT
2447            **  01/03/83   JP           Removed S9 usage
2448            **  03/09/83   JP           Changed STMBID to bSTMT
2449            **  07/05/83   JP           Save 3 levels in RSTKBF
2450            **  07/05/83   JP           Adjusted documentation
2451            **  07/22/83   NM           Moved configuration excpt handling
2452            **  09/13/83   JP           Updated documentation:
2453            **                            4 stack levels used;4 saved
2454            **
2455            ****************************************************************
2456            ****************************************************************
2457 10DC7 860   STMBF? ?ST=0  0             Coldstart ?
2458 10DCA 00          RTNYES               No.
2459 10DCC D1          B=0     A            Yes.  Size = 0
2460 10DCE 20          P=      0
2461 10DD0 3210        LC(3)   =bSTMT       Allocate Statement Buffer
           8
2462 10DD5 8C00        GOLONG  =I/OAL+      Allocate with No Leeway check
           00
2463            *
2464 10DDB 78EF =LEXBUF GOSUB  STMBF?        Create stmt buffer if necessary
2465 10DDF AF2  =LEXBF+ C=0     W
2466 10DE2 10C         R4=C                 Since not called from CONF
2467 10DE5 D1    LXBF++ B=0     A            Size=0, ID=01
2468 10DE7 20          P=      0
2469 10DE9 32CF        LC(3)   =bLEX        Buffer ID
           B
2470 10DEE 8E00        GOSUBL  =I/OAL+      Allocate with No Leeway Check
           00
2471            *
2472            * Save 4 levels on stack
2473            *
2474 10DF4 7AEB        GOSUB   R3<RST       Save 4 stack levels
2475            *
2476            * Check for LEX Files in RAM first
2477            *    Set D1 <-- Start of File Chain in RAM
2478            *    If not enough room to expand ---> go collapse down,
2479            *      add MAINT
2480            *
2481 10DF8 8E00        GOSUBL  =C=MAIN      Using D1, read MAINST into C(A)
           00
2482 10DFE 135         D1=C
2483 10E01 7C11        GOSUB   LEXFND       Find all Lex Files
2484 10E05 5A1         GONC    LEXBSP       Not enough room to expand
2485            *
2486            * Check if ROM Configuration is non-empty
2487            * For each ROM
2488            *     Search for LEX files & Update LEX table (LEXFND)
2489            *
2490 10E08 72D1        GOSUB   ROMCHK
2491 10E0C 432         GOC     LEXB30       Empty Configuartion Table
```

```
2492 10E0F 590          GONC   LEXB20          B.E.T.
2493 10E12 7912  LEXB10 GOSUB  ROMFND          Find next ROM
2494 10E16 491          GOC    LEXB30          No more ROMs
2495 10E19 7401  LEXB20 GOSUB  LEXFND          Find all LEX files in ROM
2496 10E1D 44F          GOC    LEXB10          Continue if enough room
2497             *
2498             * No room to Allocate / Expand buffer
2499             *    Collaspe Lex Buffer
2500             *    Set C(S)=1 so I/OEX2 won't use Leeway in MEMCHK
2501             *
2502 10E20 32CF  LEXBSP LC(3)  =bLEX
           B
2503 10E25 8E00         GOSUBL =I/OCOL         Collapse Lex Buffer down
           00
2504 10E2B 21           P=     1               No leeway setting
2505 10E2D 440          GOC    LEXB35          B.E.T.
2506             *
2507             * Add Built-in XROM and MAINT to LEX Table Buffer
2508             *   Set C(S)=0 so I/OEX2 will use Leeway
2509             * If not enough room to expand,  C(S) <-- 1
2510             *
2511 10E30 20    LEXB30 P=     0
2512 10E32 80FF  LEXB35 CPEX   15
2513             *
2514             * Set R3 @ "00" to indicate End of File chain for XROM01 and
2515             * MAINT.
2516             * Position D0 @ start of XROM01 (@ LEX ID field)
2517             * GOSUB at secondary entry point in LEXFND to add XROM01, MAINT
2518             *
2519 10E36 20           P=     0               P could be 1 from jump to LEXB35
2520 10E38 3400         LC(5)  =RTNSXM         Point @ "00" byte
           000
2521 10E3F 10B          R3=C                   End of file chain kludge
2522 10E42 1B00         D0=(5) =xrm01s         Start of XROM01
           000
2523 10E49 7901         GOSUB  LEXF00          Add XROM01,MAINT to Lex Buffer
2524 10E4D 52D          GONC   LEXBSP          No room to expand
2525             *
2526             * Handle special configuration problems here:
2527             *   ROMs moved
2528             *   Configuration warning
2529             *
2530 10E50 11C          C=R4
2531 10E53 812          CSLC
2532 10E56 0A           ST=C                   Restore status bits
2533 10E58 863          ?ST=0  3               System configuration changed?
2534 10E5B 91           GOYES  PUTB25          No.
2535 10E5D 84A          ST=0   10              Request CLPSTK
2536 10E60 85B          ST=1   11              Request nocat
2537 10E63 84D          ST=0   13              Clear RUNNING flag
2538 10E66 20           P=     0
2539 10E68 8E00         GOSUBL =EDITWF         Edit workfile.
           00
2540 10E6E 8E00         GOSUBL =CLOSEA         Close all files.
           00
```

```
2541 10E74 1FB7 PUTB25 D1=(5) =ESCSTA       Location of display buffer
           4F2
2542 10E7B 20          P=     0
2543 10E7D 34DD        LC(5)  (DSPMSK)+24-(ESCSTA)
           000
2544 10E84 8F00        GOSBVL =WIPOUT       Clean out display buffer
           000
2545 10E8B 8E00        GOSUBL =NOSCRL       Clear scroll bit
           00
2546 10E91 11C         C=R4
2547 10E94 812         CSLC
2548 10E97 0A          ST=C                 Restore status bits
2549 10E99 861         ?ST=0  1             Any ROMs rubbish?
2550 10E9C 01          GOYES  PUTB30        No
2551 10E9E 3300        LC(4)  =e2MROM
           00
2552 10EA4 28          P=     8
2553 10EA6 8E00        GOSUBL =MFWRN        Yes.  "e2MROM"
           00
2554 10EAC       PUTB30
2555             *
2556             * Set up for configuration poll
2557             *
2558 10EAC 1F47        D1=(5) =DSPCHX        Clear HPIL address
           6F2
2559 10EB3 D2          C=0    A
2560 10EB5 145         DAT1=C A
2561 10EB8 8E00        GOSUBL =CL-IDS        Mark buffers for deallocation
           00
2562 10EBE 8E00        GOSUBL =RS-IDS        Restore necessary system buffers
           00
2563 10EC4 108         CON(3) =bSTMT         Restore STMT buffer
2564 10EC7 808         CON(3) =bSTART                Startup
2565 10ECA 908         CON(3) =bECOMD                Ext Cmd
2566 10ECD CFB         CON(3) =bLEX                  LEX
2567 10ED0 308         CON(3) =bFIB                  File Info Buffer
2568 10ED3 408         CON(3) =bASSGN                Assign buffer
2569 10ED6 000         CON(3) 0              End of list
2570 10ED9 8E00        GOSUBL =FIBON         Restore file buffers
           00
2571             *
2572             * Look for unmarked CHARS buffer
2573             *
2574 10EDF 20          P=     0
2575 10EE1 32BF        LC(3)  (=bCHARS)-#800  ID of unmarked alt chr set bfr
           3
2576 10EE6 8E00        GOSUBL =IOFNDO        Search for it
           00
2577 10EEC 571         GONC   CNFPOL         No find.  Proceed to poll.
2578 10EEF F0          ASL    A              Low nib of size to A[1]
2579 10EF1 81C         ASRB                  Low bit of size * 8 to A[0]
2580 10EF4 90C         ?A#0   P              Odd buffersize?
2581 10EF7 D0          GOYES  CNFPOL         Yes.  Let ROM restore.
2582 10EF9 32BF        LC(3)  =bCHARS        No.  Charset buffer is pure.
           B
```

```
2583 10EFE 8E00          GOSUBL =I/ORES        Restore it.
           00
2584               *
2585               * Perform configuration poll
2586               *
2587 10F04 8E00  CNFPOL GOSUBL =FPOLL
           00
2588 10F0A BF            CON(2) =pCONFG
2589               *
2590               * Restore 4 stack levels
2591               *
2592 10F0C 7ACA          GOSUB  =RST<R3        Restore 4 stack levels
2593 10F10 831           ?XM=0                 Handled?
2594 10F13 80            GOYES  CNFP10         Yes.  Restart configuration.
2595               *
2596               * I/Obuffer cleanup
2597               *
2598 10F15 8C00  CNFP20 GOLONG =AUTDEL         Delete any buffers not restored
           00
2599 10F1B 8C2F  CNFP10 GOLONG PWCONF
           2F
2600               ********************************************************************
2601               ********************************************************************
2602               **
2603               ** Name:(S) pCONFG  -  Configuration Poll
2604               **
2605               ** Category:   POLL
2606               **
2607               ** Type:       FPOLL
2608               **
2609               ** Purpose:
2610               **       Poll at termination of configuration to allow:
2611               **         1) Claiming of I/O buffers.
2612               **         2) Changing configuration of machine.
2613               **
2614               ** Should poll be "Handled" (return with XM=0)?:
2615               **       Yes, but ONLY IF reconfiguration is desired.
2616               **
2617               ** Meaning of "Handling" Poll (what does code do if handled?):
2618               **       Calling code jumps to beginning of configuration code
2619               **       and reconfigures the system.
2620               **
2621               ** Entry conditions for handler (registers, ST, RAM, etc.):
2622               **       Carry set.
2623               **       B[A] = Poll number.
2624               **       HEX mode.
2625               **       P=0.
2626               **
2627               ** Normal exit conditions from handler if handled (ST, RAM,
2628               ** registers, etc.):
2629               **       HEX mode.
2630               **       XM=0.
2631               **
2632               ** Normal exit conditions from handler if not handled (ST, RAM,
2633               ** registers, etc.):
```

```
2634            **      HEX mode.
2635            **      XM=1.
2636            **
2637            ** Available subroutine levels:
2638            **      --FPOLL handler is two levels deeper than caller--
2639            **      LEXBUF (from where pCONF is invoked) saves 3 stack
2640            **        levels.  A responder may use UP TO 3 levels
2641            **
2642            ** NOTE:
2643            **      "Handling" the poll (returning with XM=0) is very
2644            **      serious business.  It means that you want the machine
2645            **      reconfigured.  Lazy writers of poll handlers who fail
2646            **      to RTNSXM when they should can hang the machine in
2647            **      CONFIGURATION forever.
2648            **
2649            **      --SCRATCH RAM TO CONSIDER BELOW:--
2650            **      --STMT/FN Scratch, SCRTCH, SNAPBF, TRFMBF, LDCSPC,--
2651            **      --LEXPTR.--
2652            **
2653            ** What registers/RAM may be used if handled?:
2654            **      All CPU registers.
2655            **      All scratch RAM (I think).
2656            **
2657            ** What registers/RAM may be used if not handled?:
2658            **      All CPU registers except D[A].
2659            **      All scratch RAM (I think).
2660            **
2661            ** Special memory/pointer considerations (are pointers funny?):
2662            **      May be in CALC mode.
2663            **
2664            ** Envisioned application(s):
2665            **      Three main ones: 1) claiming I/O buffers, 2) creating
2666            **      I/O buffers, and 3) changing configuration.
2667            **
2668            **      1) Claiming: This is the time to reclaim I/O buffers to
2669            **         keep them from being deleted.  Just before this
2670            **         poll, all I/O buffers are marked for deletion.  To
2671            **         keep your I/O buffers from being deleted, you need
2672            **         to perform an I/ORES on those you want to keep.
2673            **         [Marking/unmarking for deletion consists of
2674            **          clearing/setting (respectively) the upper bit of
2675            **          the buffer ID number.  Until the buffer is restored
2676            **          (unmarked), it will not be found with I/OFND
2677            **          because it will have a different number.]
2678            **
2679            **      2) Creating: This may be the time to create needed
2680            **         I/O buffers.  Or you may have done it at wakeup
2681            **         time.  Or maybe some other time.  But maybe here.
2682            **
2683            **      3) Changing: There are certain ways software can change
2684            **         the configuration of the machine; specifically by
2685            **         doing FREE or CLAIM port.  Sample situation: a
2686            **         plug-in may contain a ROM with a RAM intended only
2687            **         for the ROM's use.  When polled at configuration
2688            **         time, the ROM examines the RAM table and determines
```

```
2689            **          that the RAM living in the same plug-in is
2690            **          configured as system RAM.  The ROM then performs all
2691            **          the trappings of FREEPORT except the configuration.
2692            **          It then indicates that the poll has been handled,
2693            **          and the code reconfigures the system.  When this
2694            **          poll happens again (as it inevitably will), the ROM
2695            **          will see that its companion RAM is configured as
2696            **          IRAM, and will not repeat this monkey business.
2697            **
2698            ** History:
2699            **
2700            **    Date      Programmer            Modification
2701            **  --------    ----------   ------------------------------------
2702            **  05/11/83    NM           Added documentation
2703            **  07/05/83    JP           Added stack level usage
2704            **
2705            ********************************************************************
2706            ********************************************************************
```

```
2707                    STITLE LEXFND - Find Lang Ext Files
2708            ********************************************************************
2709            ********************************************************************
2710            **
2711            ** Name:    LEXFND  -  Find All LEX Files Within File Chain
2712            **
2713            ** Category:   CONFIG
2714            **
2715            ** Purpose:
2716            **      Find all Language Extension Files within a file chain
2717            **      Find all Language Externsion Files within one file
2718            **      Calculate Main Table start for each file
2719            **      Add LEX ID,Token Range & Main Table start to LEX Table
2720            **        Buffer.
2721            **
2722            ** Entry:
2723            **      LEXFND:  P=0
2724            **               D1 @ First file in chain
2725            **               C(S) is cleared for LEEWAY check on expand
2726            **      LEXF00:  D1 @ Lex ID of Lex file  (xrom01)
2727            **               R3 @ Next file address
2728            **                  @ "00" byte to indicate NO other files
2729            **               C(S) Set to 1 if no LEEWAY check in Buffer
2730            **                      Expand.
2731            **                      Used when no room & adding XROM01/MAINT
2732            **                    Set to 0 if LEEWAY check in Buffer Expand
2733            **
2734            ** Exit:
2735            **      P=0
2736            **      Carry set
2737            **        LEX Table updated with all Extension files in chain
2738            **      Carry clear
2739            **        Not enough memory to add to Lex Buffer
2740            **
2741            ** Calls:      RDHDR1,ASLW5,I/OEX2
2742            **
2743            ** Uses.....
2744            **  Exclusive: A,B,C,D,R2,R3,D1,D0
2745            **  Inclusive: A,B,C,D,R1,R2,R3,D1,D0
2746            **
2747            **             R2      = File Type (from RDHDR1)
2748            **             R2(B)   = LEX ID
2749            **             R2(2-5) = Token Range
2750            **             R2(6-10)= Main Table start
2751            **             R3      = Next file address
2752            **             R1      = Table information
2753            **                       used by ROMFND/LEXBUF (which calls
2754            **                       LEXFND)
2755            **             RSTK    = Addr to Next Lex File within one file
2756            **
2757            ** Stk lvls:   4
2758            **             LEXBUF (which invokds LEXFND) has saved 3 levels
2759            **             so actual usage is 1  (the CALL makes it 2)
2760            **
2761            **
```

```
2762            ** Detail:
2763            **
2764            **  LANGUAGE EXTENSION FILE FORMAT
2765            **
2766            **  Standard File Header
2767            **  LEX ID                    2 nibs
2768            **  Token Range               4
2769            **  Internal Lex File         5
2770            **  Speed Table exists?       1
2771            **  [ SPEED TABLE ]          78 (3*26)
2772            **  Speed Table exists?       1
2773            **  TEXT Table Pointer        4 nibs
2774            **  Message Table Pointer     4 nibs
2775            **  POLL Handler Pointer      5
2776            **  Main Table
2777            **  Text Table
2778            **  Message Number Range      4
2779            **  Message Table
2780            **        .
2781            **        .
2782            **
2783            **  Algorithm:
2784            **
2785            **  Repeat
2786            **    Set C(S)=0 for LEEWAY check on Buffer Expansion
2787            **    Read first 2 nibs of filename
2788            **      Repeat until (End of chain)   (Filename=0)
2789            **        Set Request File Type flag (S9=0)
2790            **        Read file header (RDHDR1)
2791            **        Set D0 = D1 (past header)
2792            **        Compute address to next file
2793            **        Save address to next file  (R3)
2794            **        If File type (R2) = Language Extension (fLEX)
2795            ** LEXF00:   Repeat until Internal Lex File chain = 0
2796            **             Read LEX ID & Token Range @ D0
2797            **             Save LEX ID,Token range,and LEEWAY setting  (B)
2798            **             Read Internal Lex File chain
2799            **             If non-zero chain
2800            **                Calculate address to next Lex File
2801            **             Save addr to next internal Lex file on stack
2802            **             Load Offset to Main Table, including Speed Table
2803            **             Read Speed Table nibble
2804            **             If a speed table doesn't exist                (=F)
2805            **                Subtract length of speed table
2806            **             Shift main table address over 6 nibs      (ASLW5)
2807            **             Save LEX ID and Token Range with Main Table (R2)
2808            **             Expand LEX Table Buffer by 11 nibs       (I/OEX2)
2809            **             If not enough memory to expand
2810            **               Pop stack level off)
2811            **               ReturnCC
2812            **             Write out LEX ID, Token Range, Main Table start
2813            **             Move LEEWAY Check setting back to C(S)
2814            **             Get next Internal Lex File chain          (RSTK)
2815            **             Set D0 @ Start of next Lex File
2816            **        Restore Next file address
```

```
2817                ** RTNSC
2818                **
2819                ** History:
2820                **
2821                **    Date     Programmer          Modification
2822                **  --------   ----------  ----------------------------------
2823                **  11/01/82   JP          Rewrote for new Lex File format
2824                **  11/04/82   JP          Add/Pack entry for XROM01/MAINT add
2825                **  01/04/82   JP          Removed S9 usage
2826                **  07/05/83   JP          Pop stk level if RTNNC from I/OEX2
2827                **
2828                ****************************************************************
2829                ****************************************************************
2830 10F21 AC2   =LEXFND C=0    S             Set Leeway Check to Expand buffer
2831 10F24 14B    LEXFD+ A=DAT1 B             Read first 2 nibs of filename
2832 10F27 968         ?A=0   B             End of Chain ?
2833 10F2A 00          RTNYES
2834                *
2835                * Read file header, Set DO past header, Compute next file
2836                * address.
2837                * Save next file address (R3)
2838                *
2839 10F2C 8F00          GOSBVL =RDHDR1       Read file header
          000
2840 10F33 137           CD1EX
2841 10F36 134           DO=C                 DO @ File Length field
2842 10F39 164           DO=DO+ lFLENh        Position past header
2843 10F3C C2            C=C+A  A             Next file=Ptr + File length
2844 10F3E 10B           R3=C                 Save next file address
2845                *
2846                * If file type = Language Extension
2847                *    Read LEX ID, Token Range and Save in B
2848                *    If address to next Lex File within one file # 0
2849                *       Compute next Lex File address
2850                *    Save next Lex File address within one file on stack
2851                *
2852 10F41 112           A=R2                 Read file type
2853 10F44 20            P=     0
2854 10F46 3480          LC(5)  =fLEX        Lang Ext File type
          2E0
2855 10F4D 8A2           ?A=C   A             Lang Ext File ?
2856 10F50 60            GOYES  LEXF00
2857 10F52 6180          GOTO   LEXF30
2858                *
2859                * Entry for Adding XROM01 and MAINT in the Lex Buffer
2860                *   R3   @ RTNSXM instruction
2861                *          Indicates End of file chain, "Next file" = 00
2862                *   D1   @ Start of XROM01 @ Lex ID
2863                *   C(S) = Leeway Setting for MEMCHK
2864                *          Default is LEEWAY checking   (C(S)=0)
2865                *
2866 10F56 15A5  LEXF00 A=DAT0 6             Read LEX ID and Token Range
2867 10F5A AF8           B=A    W            Save it
2868 10F5D 165           DO=DO+ (lLXID)+(lLXTKR)   Skip LEX ID and Token Range
2869 10F60 146           C=DAT0 A            Read next Lex file chain address
```

```
2870 10F63 8AA          ?C=0   A              End of chain ?
2871 10F66 A0           GOYES  LEXF10
2872 10F68 132          ADOEX                 Current position in file
2873 10F6B C2           C=C+A  A              Address to next lex file
2874 10F6D 132          ADOEX                 Restore DO
2875 10F70 06    LEXF10 RSTK=C                Save next Lex file address
2876 10F72 164          DO=DO+ 1LXFAD
2877                *
2878                * Load offset to Main Table, including Speed Table length
2879                * If no speed table exists  (A(S)=F)
2880                *    Subtract length of Speed Table + additional Speed Table
2881                *    nibble.
2882                * Calculate Main Table Start
2883                * Save in R2(6-10), along with LEX ID and Token Range
2884                *
2885 10F75 D2             C=0    A
2886 10F77 31D5         LC(2)  =oMAINT        Offset to Main table
2887 10F7B DA           A=C    A              Move to A
2888 10F7D 1524         A=DATO S              Read Speed Table nibble
2889 10F81 A4C          A=A-1  S              Speed Table Exists ?
2890 10F84 480          GOC    LEXF20         Yes
2891 10F87 31F4         LC(2)  (1SPDTB)+(1SPDn) Len of spd table+Spd nibble
2892 10F8B EA           A=A-C  A              Subtract speed table from offset
2893 10F8D 136   LEXF20 CDOEX                 Current position in Lex File
2894 10F90 CA           A=C+A  A              Offset to Main table
2895 10F92 8E00         GOSUBL =ASLW5         Shift over 5 nibbles
             00
2896 10F98 BF0          ASL    W              Shift over 1 more nibble
2897 10F9B 25           P=     5
2898 10F9D A94          A=B    WP             Transfer LEX ID, Token Range
2899 10FA0 20           P=     0
2900 10FA2 102          R2=A                  Save LEX ID, Tkn Rng, Mn Tbl strt
2901                *
2902                * Save Leeway Setting in B(S) for future expands
2903                * Expand LEX Table buffer to add new ID
2904                *   If not enough memory --> RTNCC
2905                * Write LEX ID, Token range, Main Table start
2906                *
2907 10FA5 D2             C=0    A
2908 10FA7 30B          LC(1)  =1LXENT        Length to Expand
2909 10FAA AF5          B=C    W              B(A)<-Length,B(S)<-Leeway Setting
2910 10FAD 32CF         LC(3)  =bLEX          LEX Table Buffer ID
             B
2911 10FB2 8E00         GOSUBL =I/OEX2        Expand with C(S)=Leeway Check
             00
2912 10FB8 07           C=RSTK                POP next file address off
2913 10FBA 500          RTNNC                 No room to expand
2914                *
2915                * Set DO at Next LEX file start (saved on RSTK)
2916                * Move Next LEX file start in D(A)
2917                * Move Expansion point in LEX buffer to C(A)
2918                * Write out LEX ID.... into buffer
2919                *
2920 10FBD 134          DO=C                  Set DO @ Next Lex file start
2921 10FC0 DF           CDEX   A              C<--exp pt in buf;D<--Nx file st
```

```
2922 10FC2 135          D1=C
2923 10FC5 112          A=R2                    Lex ID,Tkn Rng,Main table start
2924 10FC8 159A         DAT1=A 1LXENT           Write LEX ID,Tkn rng,Main Tbl st
2925              *
2926              *   Restore Leeway Check nibble
2927              *     When adding XROM01 and MAINT with NO memory left,
2928              *     LEEWAY cannot be included in the MemChk
2929              *       XROM01 and MAINT are chained together, so the second
2930              *       call to I/OEX2 must have C(S) set correctly
2931              *   Check if another LEX file is within one file
2932              *
2933 10FCC AC9          C=B    S                Move Leeway Check back to C(S)
2934 10FCF 8AF          ?D#0   A                Another Lex File ?
2935 10FD2 48           GOYES  LEXF00           Add another Lex file
2936              *
2937              * Move to next file in file chain
2938              * Assumes that
2939              *   Within any file chain, adding Lex Files uses Leeway
2940              *     Checking.
2941              *   C(S) must be reset to 0 to insure a Leeway Check is done
2942              *   Only the XROM01 and MAIN additions might use No Leeway
2943              *     checking.
2944              *     Since these Lex Files are chained, there is no "next"
2945              *       file.
2946              *
2947 10FD4 11B   LEXF30 C=R3                   Next file address
2948 10FD7 135          D1=C
2949 10FDA 664F         GOTO   LEXFND
```

```
2950                    STITLE Check for ROMs, Find File Start
2951            ********************************************************************
2952            ********************************************************************
2953            **
2954            ** Name:    ROMCHK  -  Find ROM / File Chain Start
2955            ** Name:(S) ROMFND  -  Find ROM / File Chain Start
2956            **
2957            ** Category:   GENUTL
2958            **
2959            ** Purpose:
2960            **      Check if ROMs exist
2961            **      Find file chain start within ROM/IRAM
2962            **      Return Device Information about ROM
2963            **
2964            ** Entry:
2965            **      ROMCHK:  First time entry point
2966            **        Finds ROM Configuration Table
2967            **        If non-empty, save pointers required for entry to
2968            **          ROMFND.
2969            **
2970            **      ROMFND:  Repeated entry point
2971            **        R1(X) = Length to end of Configuration Table
2972            **        R1(3-7)=Position within Configuration Table
2973            **
2974            ** Exit:
2975            **      P=0
2976            **      ROMCHK:
2977            **        Carry set:
2978            **          Empty Configuration Table
2979            **        Carry Clear:
2980            **          D1,C(A) @ First file on plug-in
2981            **          D(S) = Device type
2982            **                 1 = IRAM
2983            **                 2 = ROM
2984            **                 3 = HP EEPROM
2985            **                 4 = Intel EEPROM
2986            **                 Device type is incremented by 1
2987            **                 to distinguish from RAM
2988            **          D(0) = Port Extender #   (Device #)
2989            **          D(1) = Port #
2990            **          D(2-7) = Nibs (3-8) of config table entry
2991            **          R1(X) = Length to end of Configuration Table
2992            **          R1(3-7)=Position within Configuration Table
2993            **
2994            **          R1 must be preserved between calls to ROMFND
2995            **
2996            **      ROMFND:
2997            **        Carry set:
2998            **          No more ROMs
2999            **        Carry Clear
3000            **          Same Exit Conditions as ROMCHK
3001            **
3002            ** Calls:      CNFFND
3003            **
3004            ** Uses.......
```

```
3005            **  Exclusive: A-D,D1,R1
3006            **  Inclusive: A-D,D1,R1
3007            **
3008            ** Stk lvls:   1
3009            **
3010            ** NOTE:
3011            **      R1 must be preserved between calls to ROMFND
3012            **
3013            ** Algorithm:
3014            **
3015            ** ROMCHK:  Find ROM Configuration Table (CNFFND)
3016            **          If no table entries ---> RTNC
3017            **          Move to Device # field in table
3018            **          Move Table length to B
3019            **      1:
3020            **          Read Device#, Port# and Size infor into C,D
3021            **          Read 3 High nib address & Device type
3022            **          Adjust pointer (D1) to next entry in table
3023            **          Increment & Move Device type to D(S)
3024            **          Calculate & Read first file address
3025            **             Save Len of Config table & Next entry pos'n
3026            **             in R1.
3027            **             D1 <-- Start of file
3028            **             RTNCC
3029            ** ROMFND:  Restore Len of Config Table  (Low 3 nibs R1 -> B)
3030            **          Restore Position in Config Table  (R1 --> D1)
3031            **      2:  If entries left (B>0)
3032            **             goto 1;
3033            **          else
3034            **             RTNSC
3035            **
3036            ** History:
3037            **
3038            **    Date      Programmer              Modification
3039            **  --------   ----------   ------------------------------------
3040            **  07/09/82   JP           Modified documentation
3041            **
3042            ********************************************************************
3043            ********************************************************************
3044 10FDE 20  =ROMCHK P=     0
3045 10FE0 31EF        LC(2)  =ROMCID        ROM Configuration Table ID
3046 10FE4 74C9        GOSUB  CNFFND         Find Table
3047 10FE8 938         ?A=0   X              No table entries ?
3048 10FEB 00          RTNYES
3049 10FED 170         D1=D1+ 1              Move to Device # field
3050 10FF0 AB8         B=A    X              Move Table length to B
3051            *
3052            * Read Device# (Port Extender#) and Port# and size information
3053            * Move to Address field
3054            * Read 3 High Nib address & Device Type
3055            * Increment & Move Device Type to D(S)
3056            * Set D1,C(A) to first file in chain
3057            *
3058 10FF3 15F7  ROMF10 C=DAT1 8              Read Device#, Port #
3059 10FF7 AF7         D=C    W              Save in D(B)
```

```
3060 10FFA 172          D1=D1+ 3              Move to Address field
3061 10FFD 15F2         C=DAT1 3              Read 3 nib Address
3062 11001 172          D1=D1+ 3              Move to Device Type
3063 11004 1574         C=DAT1 S              Read Device Type
3064 11008 B46          C=C+1  S              Increment Device Type
3065 1100B AC7          D=C    S              Save in D(S)
3066 1100E 173          D1=D1+ 4              Move to next entry
3067 11011 F2           CSL    A
3068 11013 F2           CSL    A              Move address to C(4)
3069 11015 308          LCHEX  8              Low addr of file ptr
3070 11018 133          AD1EX                 Table ptr in A(A)
3071 1101B 135          D1=C                  File ptr in D1,C(A)
3072              *
3073              * Compute first file start
3074              * Save Configuration Table information
3075              * Point to first file
3076              *
3077 1101E BF0          ASL    W              Shift over Pos in Table
3078 11021 BF0          ASL    W
3079 11024 BF0          ASL    W
3080 11027 AB4          A=B    X              Save Table length
3081 1102A 101          R1=A                  Save pos'n in CNF table, len
3082 1102D 03           RTNCC
3083              *
3084              * ROMFND Entry
3085              *    Assumes:  R1(X) = Length to end of Configuration Table
3086              *                      Last returned LEX ID has NOT been
3087              *                        decremented yet.
3088              *                R1(3-7)=Position to next entry in Configuration
3089              *                        Table.
3090              *
3091 1102F 111  =ROMFND A=R1                  Restore length of Config Table
3092 11032 AB8          B=A    X
3093 11035 BF4          ASR    W              Shift Pos in table over
3094 11038 BF4          ASR    W
3095 1103B BF4          ASR    W
3096 1103E 131          D1=A                  Pos'n within Configuration Table
3097              *
3098              * Check if anymore entries in Configuration Table
3099              *
3100 11041 20           P=     0
3101 11043 32A0         LCHEX  00A            Table entry length
          0
3102 11048 B31          B=B-C  X
3103 1104B 939          ?B=0   X              No more entries ?
3104 1104E 00           RTNYES
3105 11050 52A          GONC   ROMF10         B.E.T.
3106 11053              END
```

AD1P	Abs	66368	#10340	-	808	712	714			
ASLW3	Ext			-	71					
ASLW5	Ext			-	2895					
ASRW3	Ext			-	76					
AUTDEL	Ext			-	2598					
AVMEME	Abs	193945	#2F599	-	11	2067	2069			
AVMEMS	Abs	193940	#2F594	-	11	1605	1678	1855	2069	2094
C=MAIN	Ext			-	2054	2481				
=C=RAME	Abs	66697	#10489	-	1312	55	1716	1915		
CDIV10	Abs	68979	#10D73	-	2190	1638	1957			
CDIV20	Abs	68988	#10D7C	-	2194	2199				
CL-IDS	Ext			-	2561					
CLKSPD	Ext			-	2151					
CLOSEA	Ext			-	2540					
CLRXDS	Ext			-	2146					
=CNFFND	Abs	68012	#109AC	-	1828	59	3046			
CNFND3	Abs	68019	#109B3	-	1829	1846				
CNFND5	Abs	68032	#109C0	-	1835	1832				
CNFP10	Abs	69403	#10F1B	-	2599	2594				
CNFP20	Abs	69397	#10F15	-	2598					
CNFPOL	Abs	69380	#10F04	-	2587	2577	2581			
COLD	Abs	67798	#108D6	-	1713	1698	1700			
COLD25	Abs	67459	#10783	-	1597	1600				
COLD30	Abs	67510	#107B6	-	1621	1624				
COLDS1	Abs	67379	#10733	-	1560	1558				
COLDST	Ext			-	1559					
CON130	Abs	66913	#10561	-	1399	1404				
CON140	Abs	66926	#1056E	-	1403	1407				
CON150	Abs	66963	#10593	-	1416	1419				
CON160	Abs	66974	#1059E	-	1420	1400				
CON170	Abs	67119	#1062F	-	1461	1348				
CON180	Abs	67128	#10638	-	1464	1477				
CON190	Abs	67154	#10652	-	1472	1475				
CON200	Abs	67166	#1065E	-	1476	1473				
CON210	Abs	67173	#10665	-	1478	1465				
CON220	Abs	67198	#1067E	-	1490	1511				
CON225	Abs	67218	#10692	-	1496	1499				
CON227	Abs	67230	#1069E	-	1500	1497				
CON230	Abs	67259	#106BB	-	1510	1528				
CON240	Abs	67266	#106C2	-	1512	1503				
CON250	Abs	67274	#106CA	-	1515	1517				
CON260	Abs	67289	#106D9	-	1521	1527				
CON265	Abs	67303	#106E7	-	1526	1524				
CON270	Abs	67312	#106F0	-	1529	1491				
CON275	Abs	67332	#10704	-	1536	1534				
CON280	Abs	67527	#107C7	-	1626	1556				
CON290	Abs	67566	#107EE	-	1638	1636				
CON310	Abs	67579	#107FB	-	1642	1655	1662			
CON320	Abs	67588	#10804	-	1645	1664				
CON330	Abs	67597	#1080D	-	1648	1668				
CON340	Abs	67619	#10823	-	1656	1652				
CON350	Abs	67641	#10839	-	1663	1659				
CON360	Abs	67649	#10841	-	1665	1657				
CON370	Abs	67662	#1084E	-	1669	1672				
CON380	Abs	67666	#10852	-	1670	1644				

```
 CON390  Abs   67678 #1085E -  1674  1647  1677
 CON400  Abs   67691 #1086B -  1678  1673
 CON410  Abs   67728 #10890 -  1687  1715
 CON420  Abs   67740 #1089C -  1691  1688
 CON430  Abs   67776 #108C0 -  1704  1706
 CON440  Abs   67785 #108C9 -  1708  1710
 CON450  Abs   67802 #108DA -  1714  1694  1712
 CON460  Abs   67809 #108E1 -  1716  1690
 CON470  Abs   67857 #10911 -  1735  1752  1765  1787
 CON480  Abs   67883 #1092B -  1742  1737
 CON490  Abs   67914 #1094A -  1753  1741  1745
 CON500  Abs   67952 #10970 -  1767  1759
 CON510  Abs   67964 #1097C -  1772  1770
 CON540  Abs   67994 #1099A -  1782
 CON550  Abs   68074 #109EA -  1853  1766  1774  1776  1786
 CON560  Abs   68105 #10A09 -  1863  1876
 CON570  Abs   68138 #10A2A -  1875  1888  1906
 CON580  Abs   68145 #10A31 -  1877  1868
 CON590  Abs   68179 #10A53 -  1889  1883
 CON600  Abs   68191 #10A5F -  1894  1892
 CON610  Abs   68196 #10A64 -  1896  1864
 CON650  Abs   68232 #10A88 -  1907  1896  1898
 CON660  Abs   68242 #10A92 -  1910  1914
 CON670  Abs   68258 #10AA2 -  1915  1912
 CON680  Abs   68269 #10AAD -  1919  1924
 CON685  Abs   68297 #10AC9 -  1928  1933
 CON690  Abs   68486 #10B86 -  1997  2032
 CON700  Abs   68493 #10B8D -  2000  2010
 CON710  Abs   68526 #10BAE -  2011  2008
 CON720  Abs   68591 #10BEF -  2033  2003
 CON730  Abs   68693 #10C55 -  2067  2056
 CON740  Abs   68727 #10C77 -  2078  2076
 CON750  Abs   68741 #10C85 -  2083  2086
 CON760  Abs   68765 #10C9D -  2092  2079
 CON770  Abs   68804 #10CC4 -  2102  2061
 CON775  Abs   68827 #10CDB -  2108  2093
 CON780  Abs   68854 #10CF6 -  2114  2138
 CON790  Abs   68887 #10D17 -  2126  2123
 CON800  Abs   68914 #10D32 -  2137  2143
 CON810  Abs   68937 #10D49 -  2144  2117
 CONCLD  Abs   68679 #10C47 -  2062  2072
=CONF    Abs   66066 #10212 -   649
 CONF10  Abs   66718 #1049E -  1319   679
 CONF20  Abs   66756 #104C4 -  1340  1347
 CONF55  Abs   66783 #104DF -  1348  1383
 CONF60  Abs   66787 #104E3 -  1349  1341
 CONF69  Abs   66846 #1051E -  1378  1369
 CONF70  Abs   66858 #1052A -  1382  1421
 CONFP4  Abs   66396 #1035C -   900   718   735
 CONFRS  Abs   66075 #1021B -   652  1535
=CONFS3  Abs   66069 #10215 -   650
 CONFST  Abs  195046 #2F9E6 -    11  1563  1632  1682  1828  1944  2052  2113
 CONP10  Abs   66407 #10367 -   904   922
 CONP15  Abs   66414 #1036E -   907   908
 CONP17  Abs   66429 #1037D -   912   913
```

```
 CONP20  Abs   66460 #1039C -   923   911   917
 CSLC3   Ext                 - 1447
 CSLW4   Ext                 -  687
 CSLW5   Ext                 - 1427  1449
 CSRC3   Ext                 - 1444
 CSRW3   Ext                 - 1434  1441
 CSRW5   Ext                 - 1317
 CURREN  Abs  193900 #2F56C -   11  1571
 D=AVME  Ext                 - 1681
 DSLW-P  Abs   68683 #10C4B - 2063  2034  2038  2065  2088
 DSPCHX  Abs  194164 #2F674 -   11  2558
 DSPMSK  Abs  193856 #2F540 -   11   662  2543
 EDITWF  Ext                 - 2539
 ESCSTA  Abs  193659 #2F47B -   11   658  1321  2541  2543
 FIBON   Ext                 - 2570
 FNDB20  Abs   66672 #10470 - 1271  1274
 FNDBUB  Abs   66658 #10462 - 1266  1269  1424  1428
 FPOLL   Ext                 - 2587
 I/OAL+  Ext                 - 2462  2470
 I/OCOL  Ext                 - 2503
 I/OEX2  Ext                 - 2911
 I/ORES  Ext                 - 2583
 IDL100  Abs   66345 #10329 -  738   733
 IDLOOP  Abs   66102 #10236 -  660   678   715
 IDLP10  Abs   66160 #10270 -  679   664
 IDLP20  Abs   66164 #10274 -  680   673
 IDLP30  Abs   66242 #102C2 -  707   720   737
 IDLP40  Abs   66254 #102CE -  711   707
 IDLP60  Abs   66271 #102DF -  716   697   740
 IDLP90  Abs   66288 #102F0 -  721   695
=INITPT  Abs   69013 #10D95 - 2241  1573  1587  1607  1615  2244
 INTR4   Abs  193536 #2F400 -   11    52
 IOBFEN  Abs  193910 #2F576 -   11  1585
 IOFNDO  Ext                 - 2576
=ISRAM?  Abs   65938 #10192 -   50
 ISRM10  Abs   65979 #101BB -   62    84
 ISRM20  Abs   66031 #101EF -   79    69    78
 ISRM30  Abs   66050 #10202 -   85    81
 ISRMCC  Abs   66061 #1020D -   90    54    65
 KEYDN?  Abs   66361 #10339 -  776   773   907   912
 KYDN?   Ext                 -  776
 LCFFC   Abs   66688 #10480 - 1277  1272
 LEEWAY  Abs     212 #000D4 -   12  2074
 LEXB10  Abs   69138 #10E12 - 2493  2496
 LEXB20  Abs   69145 #10E19 - 2495  2492
 LEXB30  Abs   69168 #10E30 - 2511  2491  2494
 LEXB35  Abs   69170 #10E32 - 2512  2505
=LEXBF+  Abs   69087 #10DDF - 2465
 LEXBSP  Abs   69152 #10E20 - 2502  2484  2524
=LEXBUF  Abs   69083 #10DDB - 2464
 LEXF00  Abs   69462 #10F56 - 2866  2523  2856  2935
 LEXF10  Abs   69488 #10F70 - 2875  2871
 LEXF20  Abs   69517 #10F8D - 2893  2890
 LEXF30  Abs   69588 #10FD4 - 2947  2857
 LEXFD+  Abs   69412 #10F24 - 2831
```

```
=LEXFND  Abs   69409 #10F21 -  2830  2483  2495  2949
 LOCKWD  Abs  194482 #2F7B2 -    11  1561
 LXBF++  Abs   69093 #10DE5 -  2467  2154
 MAINST  Abs  193880 #2F558 -    11  1571  1567  2099  2102
 MFWRN   Ext                -  2553
 MODS10  Abs   66590 #1041E -  1127  1129
 MODS20  Abs   66600 #10428 -  1131  1133
 MODSIZ  Abs   66579 #10413 -  1123  1346
 MOVED2  Ext                -  2101
 MOVEUA  Ext                -  2060
 MOved3  Ext                -  1781
 MRKNEW  Abs   69026 #10DA2 -  2282  1660  1665  1674
 MRKOLD  Abs   69041 #10DB1 -  2324  1661  1663  1669
=MSIZ++  Abs   66573 #1040D -  1085    72
=MSIZE   Abs   66567 #10407 -  1083  1746  1753  1869  1877
=MSIZE+  Abs   66570 #1040A -  1084
 MTHSTK  Abs  193945 #2F599 -    11  1613  1611
 Moveu3  Ext                -  1902  2021
 NOSCRL  Ext                -  2545
 PAKR40  Abs   67006 #105BE -  1429  1425
 PAKR50  Abs   67009 #105C1 -  1430  1460
 PAKR60  Abs   67044 #105E4 -  1440  1435  1437
 PAKR75  Abs   67100 #1061C -  1455  1439
 PAKR80  Abs   67110 #10626 -  1458  1443  1446
 PAKROM  Abs   66981 #105A5 -  1422  1388
 PRMPTR  Abs  193975 #2F5B7 -    11  1925
 PUTB25  Abs   69236 #10E74 -  2541  2534
 PUTB30  Abs   69292 #10EAC -  2554  2550
 PUTBUF  Abs   68423 #10B47 -  1975  1625
=PWCONF  Abs   66063 #1020F -   648  2599
 R3<RST  Abs   68066 #109E2 -  1850   651  2474
 R<Rstk  Ext                -  1851
 RAMEND  Abs  193970 #2F5B2 -    11  1613  1312
 RAWBFR  Abs  193920 #2F580 -    11  1605
 RDHDR1  Ext                -  2839
 RFADJ+  Ext                -  2107
 RFNBFR  Abs  193915 #2F57B -    11  1585
 ROMC10  Abs   68370 #10B12 -  1956  1954
 ROMC20  Abs   68379 #10B1B -  1959  1972
 ROMC30  Abs   68392 #10B28 -  1963  1968
 ROMC40  Abs   68420 #10B44 -  1973  1964  1970
 ROMC50  Abs   68423 #10B47 -  1974  1960
=ROMCHK  Abs   69598 #10FDE -  3044  2490
 ROMCID  Abs    3070 #00BFE -    12  3045
 ROMF10  Abs   69619 #10FF3 -  3058  3105
=ROMFND  Abs   69679 #1102F -  3091  2493
 ROMTPT  Abs   66613 #10435 -  1175  1353  1360  1552  1729  1943  1976  1977
                            1983  1987  1991
 RS-IDS  Ext                -  2562
 RST<R3  Abs   68058 #109DA -  1848  2152  2592
 RTNSXM  Ext                -  2520
 Rstk<R  Ext                -  1849
 SIZE10  Abs   66576 #10410 -  1086  1395  1431
 SORT    Abs   66479 #103AF -   972   996  1225  1358  1726
 SORT20  Abs   66507 #103CB -   982   987   990
```

```
 SORT40   Abs   66535 #103E7 -   991   984
 SORTP2   Abs   66477 #103AD -   971  1547  1553  1938  1981
 STMBF?   Abs   69063 #10DC7 -  2457  2153  2464
 TBLPT+   Abs   66555 #103FB -  1032  2109
 TBLPTR   Abs   66552 #103F8 -  1031  1222  1335  1352  1359  1543  1551  1626
                             1721  1727  1853  1907  1934  1942  1975  1982
 UNCFG8   Abs   66381 #1034D -   843   734   739
=WAITKY   Abs   66352 #10330 -   773   670   703   725   774  1508  1521
 WHLTBL   Abs   66644 #10454 -  1222  1334  1482  1542
 WIPOUT   Ext               -  2544
 bASSGN   Abs    2052 #00804 -    12  2568
 bCHARS   Abs    3067 #00BFB -    12  2575  2582
 bECOMD   Abs    2057 #00809 -    12  2565
 bFIB     Abs    2051 #00803 -    12  2567
 bLEX     Abs    3068 #00BFC -    12  2469  2502  2566  2910
 bSTART   Abs    2056 #00808 -    12  2564
 bSTMT    Abs    2049 #00801 -    12  2461  2563
 coldst   Abs   67367 #10727 -  1557  1555  1637  1713  1955  2062
 csrw5    Abs   66712 #10498 -  1317  1033
 csrw6    Abs   66709 #10495 -  1316  1223  2041  2111
 e2MROM   Ext               -  2551
 fLEX     Abs   57864 #0E208 -    12  2854
 lFLENh   Abs       5 #00005 -    12  2842
 lLXENT   Abs      11 #0000B -    12  2924  2908
 lLXFAD   Abs       5 #00005 -    12  2876
 lLXID    Abs       2 #00002 -    12  2868
 lLXTKR   Abs       4 #00004 -    12  2868
 lSPDTB   Abs      78 #0004E -    12  2891
 lSPDn    Abs       1 #00001 -    12  2891
 nPTR1    Abs       5 #00005 -  1571  1572
 nPTR2    Abs       2 #00002 -  1585  1586
 nPTR25   Abs       5 #00005 -  1605  1606
 nPTR3    Abs       6 #00006 -  1613  1614  1918
 oMAINT   Abs      93 #0005D -    12  2886
 pCONFG   Abs     251 #000FB -    12  2588
 xrmO1s   Ext               -  2522
```

Input Parameters

  Source file name is MN&CNF::MS

  Listing file name is MN/CNF:TI:ML::-1

  Object file name is MN%CNF:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1          *    SSS   CCC   &    FFFFF  III  L
 2          *   S   S C   C & &   F       I   L
 3          *   S     C     & &   F       I   L
 4          *    SSS  C      &    FFFF    I   L
 5          *       S C     & & & F       I   L
 6          *   S   S C   C & & & F       I   L
 7          *    SSS   CCC   && & F      III  LLLLL
 8          *
 9          *
10
11                    TITLE  File Utility & System Buffer Routines<831229.1811>
12 11053              ABS    #11053
13                    RDSYMB TI%EQU::MS
14                    RDSYMB SB%RAM::MS
```

```
                EJECT
*****************************************************************
*****************************************************************
**
** Name:(S) FTYPF#  -  Look Up File Type Given Type Number
** Name:    FTYPFD  -  Look Up File Type Given Type Number
**
** Category:   FILUTL
**
** Purpose: Searches the mainframe and LEX File type tables for
**            a given file type number.
**          A pFTYPE poll is issued to search file type table in
**          external LEX file if the mainframe file type table
**          does not contain the file type.
**
** Entry:
**          FTYPF#:
**            A(A) =  File type #  (high nib = 0)
**          FTYPFD:
**            D1 pts to file type #
**
** Exit:
**          D1 preserved
**          R0     =  D1 entry state.
**          Carry set => C(A) and B(A) point to start of entry
**                       B(S)=position of file type# within
**                            entry (1 = first filetype, etc.)
**                       A(A)  =  File type number
**          Carry clear => not found
**
** Calls:   POLL, FTBSCH
**
** Uses:
**   Exclusive:  A(A),          C, R0
**   Inclusive:  A(A),B(S),B(A),C, R0
**
** Stk Lvls:  2
**
*****************************************************************
*****************************************************************
*
*****************************************************************
*****************************************************************
**
** Name:(S) pFTYPE  -  Search for file type table entry
**
** Category:   POLL
**
** Type:       POLL
**
** Purpose:
**      Search file type table in LEX file for a given file type
**      number.
**
** Should poll be "Handled" (return with XM=0)?:Yes
```

```
                **
                ** Meaning of "Handling" Poll (what does code do if handled?):
                **      Returns with D1 pointing to the file type table entry
                **      that contains the file type.
                **
                ** Entry conditions for handler (registers, ST, RAM, etc.):
                **      B[A] = Poll number.
                **      HEX mode.
                **      P=0.
                **      A[A] = file type
                **
                ** Normal exit conditions from handler if handled (ST, RAM,
                ** registers, etc.):
                **      Carry clear
                **      HEX mode.
                **      XM=0.
                **      D1 pts to start of the file type entry in the table
                **      A(S) = Position of file type number within entry
                **             (1 = first file type, etc.)
                **      A[A] = As entry condition
                **
                ** Normal exit conditions from handler if not handled (ST, RAM,
                ** registers, etc.):
                **      Carry clear
                **      HEX mode.
                **      XM=1.
                **
                ** Error exit conditions from handler:
                **      Carry set.
                **      Hex mode.
                **      Error can only happen when there is not enough memory to
                **      do the poll at all.
                **
                ** Available subroutine levels: 4
                **
                ** What registers/RAM may be used if handled?:
                **      A-C, D1, P
                **
                ** What registers/RAM may be used if not handled?:
                **      A-C, D1, P
                **
                ** What registers/RAM may be used if error exit (POLL only)?:
                **      A-C, D1, P
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ----------------------------------
                **  04/20/83   SC           Document
                **
                ********************************************************************
                ********************************************************************
                *
11053 00    =FTYPFD A=0    A
11055 15B3          A=DAT1 4
```

```
    125 11059 137  =FTYPF# CD1EX                SAVE D1 IN R0
    126 1105C 108         R0=C
    127 1105F 1F00        D1=(5) =FTYPE
              000
    128 11066 7920        GOSUB  FTBSCH         SEARCH THE MAINFRAME TABLE
    129 1106A 4E1         GOC    FTYP10         FOUND THE ENTRY IN MAINFRAME TABLE
    130 1106D 7A46        GOSUB  poll           POLL TO SEE IF ANYBODY KNOWS IT
    131 11071 D2          CON(2) =pFTYPE
    132 11073 137         CD1EX                 GET POINTER TO C(A)
    133 11076 D5          B=C    A              COPY TO B(A)
    134 11078 AC8         B=A    S              COPY POSITION TO B(S)
    135 1107B 7A00        GOSUB  FTYP10
    136 1107F 470         GOC    FTYP05         SAY NOT FOUND IF CARRY SET
    137 11082 831         ?XM=0                 ANYBODY HANDLES IT ?
    138 11085 00          RTNYES                IF SO, SET CARRY AND RETURN
    139 11087 03   FTYP05  RTNCC                CARRY CLEAR -> NOT FOUND
    140                  *
    141 11089 118  FTYP10  C=R0                 RESTORE D1 FROM R0
    142 1108C 135         D1=C
    143 1108F D9          C=B    A
    144 11091 01          RTN
    145                  *
    146                  ************************************************************
    147                  ************************************************************
    148                  **
    149                  ** Name:(S) FTBSCH  -  Search a File Type Table by Type Number
    150                  **
    151                  ** Category:   FILUTL
    152                  **
    153                  ** Purpose:  Searches file type table by file type number.
    154                  **
    155                  ** Category: FILUTL
    156                  **
    157                  ** Entry:    A(A) = file type to search for (high nib = 0)
    158                  **           D1 points at start of table
    159                  **
    160                  ** Exit:     A(A) = entry state
    161                  **           Carry set => B(A) = pointer points to start of
    162                  **                                entry
    163                  **                       B(S) = position of filetype # within
    164                  **                              entry
    165                  **           Carry clear => not found
    166                  **
    167                  ** Uses:
    168                  **   Inclusive: B(S),B(A),C(S),C(A),D1
    169                  **
    170                  ** Calls:    None
    171                  ** Stk lvls: 0
    172                  **
    173                  ************************************************************
    174                  ************************************************************
    175                  *
    176 11093 137  =FTBSCH CD1EX
    177 11096 D5          B=C    A              SAVE ENTRY ADDRESS IN B(A)
    178 11098 137         CD1EX
```

```
 179 1109B 1577          C=DAT1 W
 180 1109F B66           C=C+1  B
 181 110A2 456           GOC    FASC25
 182 110A5 17F           D1=D1+ 16
 183 110A8 AC1           B=0    S
 184 110AB D2            C=0    A
 185 110AD 941    FTSH20 ?B=C   S              ALL TYPES IN THIS ENTRY CHECKED?
 186 110B0 3E            GOYES  FTBSCH
 187 110B2 B45           B=B+1  S              Increment position counter
 188 110B5 15F3          C=DAT1 4              READ THE TYPE NUMBER
 189 110B9 173           D1=D1+ 4
 190 110BC 8A6           ?A#C   A
 191 110BF EE            GOYES  FTSH20
 192 110C1 02            RTNSC                 IF FOUND, SET CARRY & RETURN
 193             ******************************************************************
 194             ******************************************************************
 195             **
 196             ** Name:(S) FASCFD  -  Look Up File Type Given Type Name
 197             **
 198             ** Category:   FILUTL
 199             **
 200             ** Purpose:  Search the mainframe and LEX file type tables for a
 201             **           given file type number. A pFASCH poll is issued to
 202             **           search the LEX file type table if the mainframe
 203             **           file type table does not contain the file type.
 204             **
 205             ** Entry:    D1 points at the beginning of the file type name
 206             **            which is up to five characters with trailing
 207             **            blanks.
 208             **
 209             ** Exit:     D1 past the given file type name.
 210             **           P= 0
 211             **           Carry set => A(3-0) = file type number
 212             **           Carry clear => File type not found.
 213             **
 214             ** Calls:    POLL, FILEP!, FASCH,
 215             **
 216             ** Uses:     A,B,C,R3, S10
 217             **
 218             ** Stk lvls: +3
 219             **
 220             ******************************************************************
 221             ******************************************************************
 222             *
 223             ******************************************************************
 224             ******************************************************************
 225             **
 226             ** Name:(S) pFASCH  -  Search for File Type by Name
 227             **
 228             ** Category:   POLL
 229             **
 230             ** Type:       POLL
 231             **
 232             ** Purpose:
 233             **      Search file type table in LEX file for a given file
```

```
234             **      type name.
235             **
236             ** Should poll be "Handled" (return with XM=0)?:Yes
237             **
238             ** Meaning of "Handling" Poll (what does code do if handled?):
239             **      Returns the file type number for the unprotected form
240             **      of the file type.
241             **
242             ** Entry conditions for handler (registers, ST, RAM, etc.):
243             **      B[A] = Poll number.
244             **      HEX mode.
245             **      P=0.
246             **      A[9-0]= File type in ASCII, right justified with
247             **              leading blanks(first character in A(B)).
248             **
249             ** Normal exit conditions from handler if handled (ST, RAM,
250             ** registers, etc.):
251             **      Carry clear
252             **      HEX mode.
253             **      XM=0.
254             **      A[3-0] = File type number
255             **
256             ** Normal exit conditions from handler if not handled (ST, RAM,
257             ** registers, etc.):
258             **      Carry clear
259             **      HEX mode.
260             **      XM=1.
261             **
262             ** Error exit conditions from handler (POLL only):
263             **      Carry set.
264             **      HEX mode.
265             **      Error can only happen when there is not enough memory to
266             **      do the poll at all.
267             **
268             ** Available subroutine levels: 4
269             **
270             ** What registers/RAM may be used if handled?:
271             **      A-C, D1, P
272             **
273             ** What registers/RAM may be used if not handled?:
274             **      A-C, D1, P
275             **
276             ** What registers/RAM may be used if error exit:
277             **      A-C, D1, P
278             **
279             **
280             ** History:
281             **
282             **    Date     Programmer             Modification
283             **  --------   ----------   ------------------------------------
284             **  04/20/83   SC           Document
285             **
286             ********************************************************************
287             ********************************************************************
288             *
```

```
 289 110C3 2F    =FASCFD P=     15
 290 110C5 304           LCHEX  4
 291 110C8 8F00          GOSBVL =FILEP!       READ TYPE NAME INTO A-REG
           000
 292 110CF 137           CD1EX                SAVE D1 IN R3
 293 110D2 10B           R3=C
 294 110D5 1F00          D1=(5) =FTYPE        POINTS TO START OF MAIN TABLE
           000
 295 110DC 7C10          GOSUB  FASCH         SEARCH THE MAIN TABLE
 296 110E0 431           GOC    FASC10        FOUND
 297 110E3 AF4           A=B    W             Put type back to A-reg
 298 110E6 71D5          GOSUB  poll          NOT FOUND, POLL ROM
 299 110EA C2            CON(2) =pFASCH
 300 110EC 470           GOC    FASC10        SAY NOT FOUND IF CARRY SET
 301 110EF 831           ?XM=0                ANYBODY HANDLE IT? (ADJUST CARRY)
 302 110F2 20            GOYES  FASC10
 303             *
 304 110F4 11B   FASC10  C=R3                 RESTORE D1 FROM R3
 305 110F7 135           D1=C
 306 110FA 01            RTN
 307             *
 308             ********************************************************************
 309             ********************************************************************
 310             **
 311             ** Name:    FASCH   -  Search a File Type Table by Type Name
 312             **
 313             ** Category:   FILUTL
 314             **
 315             ** Purpose: Search a file type table for a file type name
 316             **
 317             ** Entry:   A(9-0) = Type name in ASCII(right justified,
 318             **                   blank filled (first character in A(B)).
 319             **          D1 pts at start of table
 320             **
 321             ** Exit:    Carry set => A(3-0) = File type.
 322             **          Carry clear => Not found.
 323             **
 324             ** Uses:    A,B,C,D1
 325             **
 326             ** Stk lvls:  +0
 327             ********************************************************************
 328             ********************************************************************
 329             *
 330 110FC AF8   =FASCH  B=A    W             SAVE TYPE NAME IN B-REG
 331 110FF 14F   FASC20 C=DAT1 B
 332 11102 B66           C=C+1  B             END OF TABLE ?
 333 11105 540           GONC   FASC30
 334 11108 03    FASC25  RTNCC
 335 1110A 29    FASC30  P=     9
 336 1110C 174           D1=D1+ 5             POINTS TO TYPE NAME OF THE ENTRY
 337 1110F 1571          C=DAT1 WP
 338 11113 911           ?B=C   WP
 339 11116 D1            GOYES  FASC40        FOUND
 340 11118 179           D1=D1+ 10            SKIP OVER THE TYPE NAME
 341 1111B D2            C=0    A
```

```
342 1111D 15F0          C=DAT1 1            READ # OF TYPES
343 11121 C6            C=C+C  A            MULTIPLY IT BY 4
344 11123 C6            C=C+C  A
345 11125 133           AD1EX
346 11128 CA            A=A+C  A
347 1112A 131           D1=A                POINTS TO NEXT ENTRY
348 1112D 170           D1=D1+ 1
349 11130 5EC           GONC   FASC20       (B.E.T.)
350 11133 17A   FASC40  D1=D1+ 11           POINTS TO 1ST TYPE IN THIS ENTRY
351 11136 15B3          A=DAT1 4            READ THE TYPE TO A
352 1113A 20            P=     0
353 1113C 02            RTNSC
354             *
355             *
```

```
356                     STITLE Search channel # in Assign Table
357                *
358                **********************************************************************
359                **********************************************************************
360                **
361                ** Name:    GETARG  -  Get arguments from program line.
362                **
363                ** Category:   EXCUTL
364                **
365                ** Purpose: Get one or two numeric arguments from a program
366                **          line.
367                **
368                ** Entry:   D0 points at the terminator preceds the argument.
369                **          (May be a comma token or end-of-line token).
370                **
371                ** Exit:    R2(A) = 1st argument(sero if no argument found)
372                **          R3(A) = 2nd argument(zero if onle one argument)
373                **
374                ** Calls:   EXPEX-, CHKEOL, POPARG
375                **
376                ** Uses:    Everything except the STMTR0 & STMTR1
377                **          (expression execution is called)
378                **
379                ** Stk lvls: +5
380                **
381                **********************************************************************
382                **********************************************************************
383                *
384 1113E 8E00 =GETARG GOSUBL =CHKEOL        CHECK IF AT END OF LINE
          00
385 11144 504          GONC   NOARG        IF SO, NO ARGUMENT FOUND
386 11147 161          D0=D0+ 2
387 1114A 8E00         GOSUBL =EXPEX-
          00
388 11150 8E00         GOSUBL =CHKEOL       SEE IF AT EOL AFTER 1ST ARGUMENT
          00
389 11156 582          GONC   ONEARG        IF SO, ONLY ONE ARGUMENT FOUND
390 11159 161          D0=D0+ 2
391 1115C 8E00         GOSUBL =EXPR
          00
392 11162 7820         GOSUB  POPARG
393 11166 17F          D1=D1+ 16            DROP THE STACK
394 11169 AF2  GARG10  C=0    W
395 1116C D6           C=A    A
396 1116E 10B          R3=C                R3= 2ND ARGUMENT
397 11171 7910         GOSUB  POPARG
398 11175 AF2  GARG30  C=0    W
399 11178 D6           C=A    A
400 1117A 10A          R2=C                R2 = 1ST ARGUMENT
401 1117D 03           RTNCC
402 1117F AF0  ONEARG  A=0    W
403 11182 56E          GONC   GARG10
404 11185 AF0  NOARG   A=0    W
405 11188 103          R3=A
406 1118B 59E          GONC   GARG30
```

```
407                *
408                ****************************************************************
409                ****************************************************************
410                **
411                ** Name:    POPARG  -  Pop and Convert Argument to Binary
412                **
413                ** Category:   EXCUTL
414                **
415                ** Purpose:  Pop a numeric number off the math stack and convert
416                **           it to binary
417                **
418                ** Entry:    D1 points at top of the math stack.
419                **
420                ** Exit:     A(A) = Binary of the argument.
421                **            Error exit to error routine if argument greater
422                **            than 5 hex digits or negative number or complex
423                **            number.
424                **
425                ** Calls: POP1N, FLTDH
426                **
427                ** Uses:  A, B, C, XM
428                **
429                ** Stk lvls: +1
430                **
431                **
432                ****************************************************************
433                ****************************************************************
434 1118E 8E00 =POPARG GOSUBL =POP1N
           00
435 11194 04           SETHEX
436 11196 4F0          GOC    CMPLER
437 11199 8E00         GOSUBL =fltdh
           00
438 1119F 400          RTNC
439                * NEGATIVE NUMBER OR HUGE NUMBER SAY "OUT OF RANGE"
440                *
441 111A2 6EA2         GOTO   OUTRNG
442
443 111A6 8C00  CMPLER GOLONG =RDATTY
           00
444
445                *CMPLER LC(2)  =eDATTY
446                *       GOTO   ERREX
447                *
448                ****************************************************************
449                ****************************************************************
450                **
451                ** Name:(S) pSREC#  -  Position to Rec# of File w/Copycode 8
452                **
453                ** Category:   POLL
454                **
455                ** Type:       POLL
456                **
457                **
458                ** Purpose:
```

```
459             **      Set file pointer to a given record # of a file whose
460             **      copy code is >= 8.
461             **
462             ** Should poll be "Handled" (return with XM=0)?:Yes
463             **
464             ** Meaning of "Handling" Poll (what does code do if handled?):
465             **      Set the file pointer in FIB to a given record # in the
466             **      file.
467             **
468             ** Entry conditions for handler (registers, ST, RAM, etc.):
469             **      B[A] = Poll number.
470             **      HEX mode.
471             **      P=0.
472             **      A[s] =  copy code of the file
473             **      A[4-0] = FIB entry address of the file
474             **      STMTD1 = FIB entry address of the file
475             **      R1 = Record # (first record of the file is record 0)
476             **
477             ** Normal exit conditions from handler if handled (ST, RAM,
478             ** registers, etc.):
479             **      Carry clear.
480             **      HEX mode.
481             **      XM=0.
482             **      Following field in FIB of the file is updated:
483             **        .Current position set to start of the given record.
484             **        .# of bytes left in current is set to equal to record
485             **         length.
486             **        .If the file is in external device, the file I/O
487             **         buffer should contain the current sector.
488             **
489             ** Normal exit conditions from handler if not handled (ST, RAM,
490             ** registers, etc.):
491             **      Carry clear (POLL only).
492             **      HEX mode.
493             **      XM=1.
494             **
495             ** Error exit conditions from handler (POLL only):
496             **      Carry set.
497             **      HEX mode.
498             **
499             ** Available subroutine levels:
500             **
501             ** What registers/RAM may be used if handled?:
502             **      All CPU registers
503             **      Don't use STMTD0 & STMTD1
504             **
505             ** What registers/RAM may be used if not handled?:
506             **      All CPU registers
507             **      Don't use STMTD0, STMTD1, R1
508             **
509             ** What registers/RAM may be used if error exit ?:
510             **      All CPU registers
511             **
512             ** History:
513             **
```

```
514             **    Date     Programmer              Modification
515             **  --------   ----------   ----------------------------------
516             **  04/20/83   SC           Document
517             **
518             ********************************************************************
519             ********************************************************************
520             *
521             ********************************************************************
522             ********************************************************************
523             **
524             ** Name:    GETRE#  -  Position Open File to Record
525             ** Name:    SREC#A  -  Move file pointer to a given record
526             **
527             ** Category:   FILUTL
528             **
529             ** Purpose:  Move the file pointer to a given record #
530             **           (The current file pointer is stored in FIB)
531             **
532             ** Entry:
533             **
534             **   SREC#A:  The record number is in A-Reg
535             **            STMTD1(4-0) = FIB entry address of the file
536             **
537             **   GETRE#:  This entry point is for RESTORE#, PRINT#, READ#
538             **            DO pts past the channel number
539             **            A= Channel #
540             **
541             ** Exit:  S9 = 0 If serial access
542             **           = 1 If Random access
543             **         STMTDO = Current PC (past the record number)
544             **         The current position in the FIB will be updated
545             **         Error exit on following conditions:
546             **          1. Record # does not exist
547             **          2. File type unrecognized
548             **   SREC#A:
549             **         DO of entry condition is saved in STMTDO
550             **
551             ** Calls: EXPEX-, POPARG, FIBADR, POLL, A-MULT
552             **
553             ** Uses:
554             **   GETRE#:
555             **         Use all registers.
556             **         Everything except STMTR0, STMTR1, STMTDO, STMTD1
557             **   SREC#A:
558             **         A,B,C,D,R1,D1,DO, ST 0-4
559             **
560             ** Stk lvls:
561             **   SREC#, GETRE#:  +5
562             **   SREC#S, SREC#A: +2
563             ********************************************************************
564             ********************************************************************
565             *
566             I/Obuf  EQU    10
567             Random  EQU    9
568             *
```

```
569 111AC 161  =RESTR# D0=D0+ 2            JUMP OVER THE \#\
570 111AF 7472        GOSUB  GETCH#
571 111B3 7F00        GOSUB  GETRE#
572 111B7 879         ?ST=1  Random        HAS RECORD # SPECIFIED ?
573 111BA 80          GOYES  RES#10       IF SO, DONE
574           * RECORD # NOT SPECIFIED IN THE RESTORE # STATEMENT, DEFAULT
575           * TO RECORD #0.
576           *
577 111BC D0          A=0    A
578 111BE 7030        GOSUB  SREC#A
579           *
580 111C2 6B94  RES#10 GOTO  CTE300
581           *
582           *
583 111C6 7D82 =GETRE# GOSUB  FIBADR
584 111CA 849         ST=0   Random        ASSUME IS A SERIAL PRINT OR READ
585 111CD 14A         A=DAT0 B
586 111D0 3100        LC(2)  =tEOL
587 111D4 962         ?A=C   B
588 111D7 00          RTNYES
589 111D9 3100        LC(2)  =tCOMMA
590 111DD 966         ?A#C   B
591 111E0 00          RTNYES
592 111E2 161         D0=D0+ 2
593           *
594           *
595 111E5 8E00        GOSUBL =EXPEX-
          00
596 111EB 7F9F        GOSUB  POPARG
597           *
598 111EF 859         ST=1   Random        SET RANDOM FLAG
599           *
600 111F2 101   SREC#A R1=A                SAVE RECORD # IN R1
601           *
602 111F5 71A2        GOSUB  SAVEDO        SAVE D0 IN STMTD0
603 111F9 1F69        D1=(5) =STMTD1       GET FIB ENTRY ADDRESS
          8F2
604 11200 143         A=DAT1 A
605 11203 131         D1=A                 D1 @ FIB OF THE FILE
606 11206 179         D1=D1+ (oCOPYb)      READ THE FILE COPY CODE FROM FIB
607 11209 1534        A=DAT1 S             A(S)= FILE COPY CODE
608 1120D 171         D1=D1+ (oDEVCb)-(oCOPYb)
609 11210 147         C=DAT1 A             C(0)= DEVICE TYPE OF THE FILE
610 11213 84A         ST=0   I/Obuf
611 11216 A06         C=C+C  P             SEE IF FILE IN EXTERNAL DEVICE
612 11219 591         GONC   SRE#30
613 1121C 85A         ST=1   I/Obuf
614 1121F 431         GOC    SRE#30
615           *
616           * POLL FOR EXTERNAL FILE OR UNRECOGNIZED FILE TYPE
617           *  ENTRY:  A(S)= FILE COPY CODE
618           *          A(A)= FIB ENTRY ADDRESS
619           *          STMTD1(4-0)= FIB ENTRY ADDRESS
620           *          R1= RECORD #
621           *  EXIT:   RETURN TO CALLING ROUTINE WITH:
```

```
622                *          CURRENT POSITION IN FIB IS UPDATED.
623                *          IF FILE IS IN TAPE, MOVE THE TAPE TO MATCH THE
624                *            CURRENT POSITION.
625                *
626 11222 7594  SRE#PL GOSUB  poll
627 11226 82           CON(2) =pSREC#
628 11228 452          GOC    SRE#35          ERROR
629 1122B 831          ?XM=0                  ANY BODY HANDLE IT ?
630 1122E 00           RTNYES                 IF SO, RETURN
631 11230 5D1          GONC   SRE#35          SAY "Illegal file type"
632                * A = FIB ENTRY ADDRESS
633 11233 D2    SRE#30 C=0    A
634 11235 3151         LC(2)  =oDBEGb
635 11239 C2           C=A+C  A
636 1123B 135          D1=C
637 1123E 147          C=DAT1 A               READ DATA START ADDRESS
638 11241 D7           D=C    A               D= DATA START ADDRESS
639 11243 119          C=R1
640 11246 AC6          C=A    S               C(S) = COPY CODE
641 11249 94C          ?A#0   S               IS THIS A DATA FILE ?
642 1124C 60           GOYES  SRE#40
643 1124E 6022  SRE#35 GOTO   BADTYP
644                *
645 11252 A46   SRE#40 C=C+C  S               COPY CODE >= 8 ?
646 11255 4CC          GOC    SRE#PL          IF SO, POLL
647 11258 A46          C=C+C  S               COPY CODE = 4 ?
648 1125B 447          GOC    LIF1R#          IF SO, IS A LIF1 ASCII FILE
649 1125E A46   SRE#41 C=C+C  S               COPY CODE = 2 ?
650 11261 4A4          GOC    HP41R#          IF SO, IS A HP-41 DATA FILE
651                * THIS IS THE FIXED LENGTH DATA FILE
652                * D1 @ DATA START OF FIB
653                * C= RECORD #
654                * D= DATA START ADDRESS
655 11264 17A          D1=D1+ (oREC#b)-(oDBEGb)
656 11267 D0           A=0    A
657 11269 15B3         A=DAT1 4               A= FILE LENGTH IN # OF RECORDS
658                *
659 1126D 8B6   TIT#10 ?C<A   A               RECORD # EXIST ?
660 11270 F0           GOYES  TIT#20
661 11272 8AA          ?C=0   A               WANT TO GOTO RECORD 0 ?
662 11275 A0           GOYES  TIT#20
663                *
664 11277 3100 =ENDFIL LC(2)  =eEOFIL
665 1127B 6D33         GOTO   ERREX
666                *
667 1127F 173   TIT#20 D1=D1+ (oRECLb)-(oREC#b)
668 11282 15B3         A=DAT1 4               A=RECORD LENGTH IN BYTES
669 11286 7C55         GOSUB  a-mult
670 1128A D2           C=0    A
671 1128C 15F3         C=DAT1 4               C= RECORD LENGTH IN BYTES
672 11290 173          D1=D1+ (oCPOSb)-(oRECLb)
673 11293 17B   TIT#30 D1=D1+ (oRLENb)-(oCPOSb)
674 11296 145          DAT1=C A               WRITE BYTES LEFT IN CURRENT RECORD
675 11299 7701         GOSUB  STFPTR          SET CURRENT POSITION
676                *
```

```
677                 * Restore D0 from STMTD0
678                 *
679 1129D 1B19 =RESTD0 D0=(5) =STMTD0
          8F2
680 112A4 146           C=DATO A
681 112A7 134           D0=C
682 112AA 03            RTNCC
683                 * SDATA FILE
684                 *  D1 @ DATA START ADDRESS
685                 *  C= RECORD #
686                 *  D= DATA START
687 112AC 17E    HP41R# D1=D1+ (oRECLb)-(oDBEGb)
688 112AF 179           D1=D1+ (oDLENb)-(oRECLb)
689 112B2 AF0           A=0    W
690 112B5 143           A=DAT1 A             A(A) = FILE DATA LENGTH IN NIBS
691 112B8 81C           ASRB                 A(A) = FILE DATA LENGTH IN BYTES
692                 *
693 112BB C6            C=C+C  A             8 BYTES PER RECORD
694 112BD C6            C=C+C  A
695 112BF C6            C=C+C  A
696 112C1 DE            ACEX   A             C= FILE LENGTH, A= RECORD POSITION
697 112C3 8BE           ?A>=C  A             PAST END OF FILE YET ?
698 112C6 1B            GOYES  ENDFIL
699 112C8 1C5    H41#10 D1=D1- (oDLENb)-(oCPOSb)
700 112CB D2            C=0    A
701 112CD 55C           GONC   TIT#30        (B.E.T.)
702                 * TEXT DATA FILE
703                 *  D1 @ DATA START OF FIB
704                 *  C= RECORD #
705                 *  D = DATA START
706 112D0 109    LIF1R# R1=C                 R1= RECORD NUMBER
707 112D3 86A           ?ST=0  I/Obuf
708 112D6 81            GOYES  LIF#05
709 112D8 D0            A=0    A
710 112DA 76C0          GOSUB  STFPTR        SET CURRENT POS TO START OF FILE
711 112DE 8E00          GOSUBL =FNDFBF       READ THE 1ST REC TO FILE I/O BUFR.
          00
712 112E4 AF2           C=0    W
713 112E7 D9            C=B    A
714 112E9 D7            D=C    A             D(A)= I/O BUFR START ADDRESS
715 112EB 108           R0=C
716 112EE 17E    LIF#05 D1=D1+ (oRECLb)-(oDBEGb)
717 112F1 179           D1=D1+ (oDLENb)-(oRECLb)
718 112F4 AF2           C=0    W
719 112F7 147           C=DAT1 A             C(A) = FILE DATA LENGTH IN NIBS
720 112FA 81E           CSRB                 C(A) = FILE DATA LENGTH IN BYTES
721 112FD 129           CR1EX                R1 = FILE LENGTH IN BYTES
722 11300 D5            B=C    A             B(A) = RECORD NUMBER
723 11302 DB            C=D    A
724 11304 D3            D=0    A             D= CURRENT POSITION (RELATIVE)
725 11306 134    LIF#07 D0=C                 D0 @ DATA START
726 11309 119    LIF#10 C=R1                 C= FILE LENGTH
727 1130C 8B7           ?C>D   A             PAST END OF FILE YET ?
728 1130F C0            GOYES  LIF#40        IF NOT, GOTO LIF#40
729                 *
```

```
730 11311 20    LIF#30 P=     0
731 11313 DB           C=D    A
732 11315 DA           A=C    A
733 11317 60BF         GOTO   H41#10
734             *
735 1131B 8E00 LIF#40  GOSUBL =RDLNAS        READ STRING LENGTH TO A
          00
736 11321 23           P=     3             A= RECORD LENGTH
737 11323 B14          A=A+1  WP            REACHED END OF FILE ?
738 11326 4AE          GOC    LIF#30
739 11329 CD           B=B-1  A             REACHED THE TARGET RECORD # ?
740 1132B 45E          GOC    LIF#30
741 1132E A1C          A=A-1  WP
742 11331 E7           D=D+1  A
743 11333 E7           D=D+1  A
744 11335 D2           C=0    A
745 11337 E6           C=C+1  A
746 11339 0E62         C=C&A  B
747 1133D 96A          ?C=0   B             RECORD LENGTH EVEN NUMBER ?
748 11340 50           GOYES  LIF#50        IF SO, GOTO LIF#50
749 11342 B14          A=A+1  WP            ADD ONE BYTE IF ODD NUMBER LENGTH
750 11345 D6    LIF#50 C=A    A
751 11347 C3           D=D+C  A             UPDATE CURRENT POSITION
752 11349 20           P=     0             TWO BYTES OF RECORD LENGTH
753 1134B 86A          ?ST=0  I/Obuf
754 1134E C0           GOYES  LIF#60
755 11350 8E00         GOSUBL =SKPBYT       SKIP OVER THE RECORD
          00
756 11356 62BF         GOTO   LIF#10
757             *
758 1135A C6    LIF#60 C=C+C  A             RECORD LENGTH IN NIBBLES
759 1135C 132          ADOEX
760 1135F C2           C=A+C  A
761 11361 64AF         GOTO   LIF#07
762             **
763             *
764             ********************************************************************
765             ********************************************************************
766             **
767             ** Name:(S) REWIND  -  Rewind Open File
768             **
769             ** Category:   FILUTL
770             **
771             ** Purpose:  Set the current position in the FIB to start of
772             **           of data in a file.
773             **
774             ** Entry:
775             **       A      =  FIB entry address of the file
776             **
777             ** Exit:
778             **       A(B)   =  FIB # of the file
779             **       STMTD1 =  FIB entry address of the file
780             **     Carry set => successful
781             **     Never returns if HP-IL error happens, exit to MFERR.
782             **
```

```
783             ** Calls:   STFPTR
784             **
785             ** Uses:  A, B, C, D, D1, D0, S10, STMTD1, S4-0
786             **
787             ** Stk lvls:  1 - internal file
788             **           4 - external file
789             ********************************************************************
790             ********************************************************************
791             *
792 11365 1F69 =REWIND D1=(5) =STMTD1        SAVE FIB ENTRY ADDRESS IN STMTD1
        8F2
793 1136C 141           DAT1=A A
794 1136F 131           D1=A
795 11372 171           D1=D1+ (oFBF#b)
796 11375 147           C=DAT1 A
797 11378 D0            A=0    A
798 1137A 84A           ST=0   I/Obuf          ASSUME IS INTERNAL FILE
799 1137D 93A           ?C=0   X
800 11380 E0            GOYES  RWND10
801 11382 85A           ST=1   I/Obuf
802 11385 17A           D1=D1+ (oFBEGb)-(oFBF#b)
803 11388 175           D1=D1+ (oSHLNb)-(oFBEGb)
804 1138B 14B           A=DAT1 B               A(B) = SUBHEADER LENGTH
805 1138E 7210  RWND10  GOSUB  STFPTR
806 11392 1F69          D1=(5) =STMTD1         Fetch FIB#
        8F2
807 11399 143           A=DAT1 A                .
808 1139C 131           D1=A                    .
809 1139F 143           A=DAT1 A                .
810 113A2 02            RTNSC
811             *
812             ********************************************************************
813             ********************************************************************
814             **
815             ** Name:    STFPTR  -  Position Open File
816             ** Name:    STFPT-  -  Set current file position
817             **
818             ** Category:   FILUTL
819             **
820             ** Purpose:  Set the current position of an opened file to a
821             **           given position
822             **
823             ** Entry:  STMTD1 = Contains the FIB entry address of the file.
824             **         S10 = 1, if file in an external device
825             **               0, if file in RAM/ROM
826             **         A(A) = New file position in bytes
827             **
828             ** Exit:  Carry clear => No error
829             **        Never returns if HP-IL error happens, exits to MFERR.
830             **
831             ** Calls:  FPOLL(pWRCBF & pRDCBF)
832             **
833             ** Uses:  A, B, C, D, D0, S4-0
834             **
835             ** Stk lvls:  +3
```

```
836           ********************************************************************
837           ********************************************************************
838           *
839           *
840           ********************************************************************
841           ********************************************************************
842           **
843           ** Name:(S) pRDCBF  -  Read Current Sector From Mass Memory
844           **
845           ** Category:   POLL
846           **
847           ** Type:       FPOLL
848           **
849           ** Purpose:
850           **    Using the FIB, read the current sector of a file in
851           **    the mass memory device into the I/O buffer that is
852           **    associated with the file.
853           **
854           **    There are total of 3 polls can be used to read/write a
855           **    sector between a mass memory device and I/O buffer:
856           **    1. pRDNBF - Write buffer out to current sector and read
857           **                in next sector. If buffer content has not been
858           **                altered, just read in next sector.
859           **    2. pRDCBF - Read in current sector from mass memory device
860           **                to the I/O buffer. This poll does not care
861           **                about the content currently in the I/O buffer.
862           **    3. pWRCBF - Write I/O buffer to current sector in the mass
863           **                memory device.
864           **
865           **
866           ** Should poll be "Handled" (return with XM=0)?:Yes
867           **
868           ** Meaning of "Handling" Poll (what does code do if handled?):
869           **      Read the current sector from the mass memory device into
870           **      the I/O buffer of the file.
871           **
872           ** Entry conditions for handler (registers, ST, RAM, etc.):
873           **      B[A] = Poll number.
874           **      HEX mode.
875           **      P=0.
876           **      STMTD1 contains the FIB entry address of the file
877           **      (SNAPBF contains A, D, D0 and D1 to restore on exit)
878           **
879           ** Normal exit conditions from handler if handled (ST, RAM,
880           ** registers, etc.):
881           **      Carry clear
882           **      HEX mode.
883           **      XM=0.
884           **      File access nib in FIB is set to zero.
885           **      A,D,D0,D1 restored to values from SNAPBF.
886           **
887           **      This poll handler must always call the routine SNAPRS to
888           **       restore A,D,D0,D1 from the snap save RAM before return.
889           **      So if the polling routine calls the SNAPSV before
890           **       issuing this poll, it can consider A,D,D0,D1 will not
```

```
891           **       be changed by this poll handler.
892           **
893           ** Normal exit conditions from handler if not handled (ST, RAM,
894           ** registers, etc.):
895           **      HEX mode.
896           **      XM=1.
897           **
898           ** Error exit conditions from handler (POLL only):
899           **      Won't return to calling routine if error occur, direct
900           **       exit to BSERR routine.
901           **
902           ** Available subroutine levels:
903           **      3
904           **
905           ** What registers/RAM may be used if handled?:
906           **      A-D, D0, D1, P, ST[0-4]
907           **      (B,C,P,ST[0-4] if SNAPSV was called)
908           **
909           ** What registers/RAM may be used if not handled?:
910           **      B, C, D0
911           **      (SNAPRS is not called)
912           **
913           ** History:
914           **
915           **    Date     Programmer             Modification
916           **  --------   ----------   ----------------------------------
917           **  04/20/83   SC           Document
918           **
919           ****************************************************************
920           ****************************************************************
921           *
922           ****************************************************************
923           ****************************************************************
924           **
925           ** Name:(S) pWRCBF  -  Write I/O Buffer to Mass Memory Device
926           **
927           ** Category:   POLL
928           **
929           ** Type:       FPOLL
930           **
931           ** Purpose:
932           **    Using the FIB, write the file I/O buffer to
933           **    the sector it came from in a mass memory device. Buffer
934           **    content, current position and record address are not
935           **    changed by this operation.
936           **
937           **    There are total of 3 polls can be used to read/write a
938           **    sector between a mass memory device and I/O buffer:
939           **    1. pRDNBF - Write buffer out to current sector and read
940           **                in next sector. If buffer content has not been
941           **                altered, just read in next sector.
942           **    2. pRDCBF - Read in current sector from mass memory device
943           **                to the I/O buffer. This poll does not care
944           **                about the content currently in the I/O buffer.
945           **    3. pWRCBF - Write I/O buffer to current sector in the mass
```

```
946             **              memory device.
947             **
948             **
949             ** Should poll be "Handled" (return with XM=0)?:Yes
950             **
951             ** Meaning of "Handling" Poll (what does code do if handled?):
952             **     Write the file I/O buffer to the sector it came from in a
953             **     mass memory device.
954             **
955             ** Entry conditions for handler (registers, ST, RAM, etc.):
956             **      B[A] = Poll number.
957             **      HEX mode.
958             **      P=0.
959             **      STMTD1 contains the FIB entry address of the file
960             **      (SNAPBF contains A,D1,D0 and D1 to restore on exit)
961             **
962             ** Normal exit conditions from handler if handled (ST, RAM,
963             ** registers, etc.):
964             **      Carry clear (POLL only).
965             **      HEX mode.
966             **      XM=0.
967             **      File access nib in FIB is set to zero.
968             **      A,D,D0,D1 restored to value from SANPBF.
969             **
970             **      This poll handler must always call the routine SNAPRS to
971             **       restore A,D,D0,D1 from the snap save RAM before return.
972             **      So if the polling routine calls the SNAPSV before
973             **       issuing this poll, it can consider A,D,D0,D1 will not
974             **       be changed by this poll handler.
975             **
976             ** Normal exit conditions from handler if not handled (ST, RAM,
977             ** registers, etc.):
978             **      HEX mode.
979             **      XM=1.
980             **
981             ** Error exit conditions from handler
982             **      Won't return to calling routine if error occurs, direct
983             **       exit to BSERR routine.
984             **
985             ** Available subroutine levels:
986             **      3
987             **
988             ** What registers/RAM may be used if handled?:
989             **      A-D, D0, D1, P, ST[0-4]
990             **      (B,C,P,ST[0-4] if SNAPSV been called)
991             **
992             ** What registers/RAM may be used if not handled?:
993             **      B, C, D0
994             **      (SNAPRS is not called)
995             **
996             ** History:
997             **
998             **    Date     Programmer             Modification
999             **  --------   ----------   ------------------------------------
1000            **  04/20/83   SC           Document
```

```
1001                **
1002                ************************************************************************
1003                ************************************************************************
1004                *
1005                *
1006                *
1007 113A4 C4    =STFPTR A=A+A  A              TURN BYTES INTO NIBBLES
1008 113A6 D6            C=A    A
1009 113A8 D7            D=C    A
1010 113AA 8E00          GOSUBL =DO=FIB
          00
1011 113B0 D5            B=C    A
1012 113B2 169           DO=DO+ (oCOPYb)
1013 113B5 1564          C=DATO S              C(S)= COPY CODE
1014 113B9 16E           DO=DO+ (oDBEGb)-(oCOPYb)+4
1015 113BC 16E           DO=DO+ (oCPOSb)-(oDBEGb)-4  DO @ CURRENT POSITION IN FIB
1016 113BF 87A           ?ST=1  I/Obuf
1017 113C2 90            GOYES  SFPT20
1018 113C4 DB    SFPT10  C=D    A
1019 113C6 144           DATO=C A              WRITE NEW POSITION
1020 113C9 03    SFPT15  RTNCC
1021                *
1022 113CB 146   SFPT20   C=DATO A              READ THE CURRENT POSITION
1023 113CE 25            P=     5
1024 113D0 A80           A=0    P
1025 113D3 A82           C=0    P
1026 113D6 20            P=     0
1027 113D8 81E           CSRB
1028 113DB 81C           ASRB
1029                *
1030 113DE AE0           A=0    B
1031 113E1 AE2           C=0    B
1032 113E4 8A2           ?A=C   A              STILL IN THE SAME RECORD ?
1033 113E7 DD            GOYES  SFPT10         IF SO, GOTO SFPT30
1034                *
1035 113E9 D9            C=B    A
1036 113EB 134           DO=C
1037 113EE 16A           DO=DO+ (oACCSb)
1038 113F1 14E           C=DATO B
1039 113F4 90A           ?C=0   P              ACCESS = READ ?
1040 113F7 E0            GOYES  SFPT40         IF SO, DON'T WRITE THE RECORD BACK
1041 113F9 07            C=RSTK                Save one level of RSTK
1042 113FB 7AF6          GOSUB  fpoll+         SAVE A,D,DO,D1 & FAST POLL
1043 113FF A1            CON(2) =pWRCBF        WRITE CURRENT RECORD BACK FIRST
1044 11401 D9            C=B    A
1045 11403 06            RSTK=C                Restore one level of RSTK
1046                *
1047 11405 8E00  SFPT40  GOSUBL =DO=FIB
          00
1048 1140B 16F           DO=DO+ (oFBEGb)+3
1049 1140E 16F           DO=DO+ (oREC#b)-(oFBEGb)-3
1050 11411 167           DO=DO+ (oCPOSb)-(oREC#b)
1051 11414 DB            C=D    A
1052 11416 144           DATO=C A              WRITE NEW POSITION
1053 11419 07            C=RSTK                Save one level of RSTK
```

```
1054 1141B 7AD6          GOSUB  fpoll+          FAST POLL
1055 1141F 81            CON(2) =pRDCBF         READ IN THE CURRENT RECORD
1056 11421 D9            C=B    A
1057 11423 06            RSTK=C                 Restore one level of RSTK
1058 11425 03            RTNCC
1059                *
1060                ************************************************************
1061                ************************************************************
1062                **
1063                ** Name:(S) GETCH#  -  Get Channel Number
1064                **
1065                ** Category:   EXCUTL
1066                **
1067                ** Purpose: Get the Given channel for a statement
1068                **
1069                ** Entry:   DO points at the channel number token.
1070                **
1071                ** Exit:    A(B) = Channel number in binary
1072                **           DO past channel number
1073                **           CHN#SV = Channel #
1074                **           Error exit if channel # > 255 or <= 0
1075                **
1076                ** Uses:    All CPU registers, status. scratch RAM except
1077                **          All scratch RAM except STMTR0, STMTR1
1078                **          (Expression execution is called)
1079                **
1080                ** Calls: EXPR
1081                **
1082                ** Stk lvls: +5
1083                ************************************************************
1084                ************************************************************
1085                *
1086 11427 8F00 =GETCH# GOSBVL =D1FSTK
             000
1087 1142E 8E00 =GETC#- GOSUBL =EXPR
             00
1088 11434 765D          GOSUB  POPARG
1089 11438 1FF6          D1=(5) =CHN#SV
             9F2
1090 1143F 149           DAT1=A B
1091 11442 D2            C=0    A
1092 11444 A6E           C=C-1  B
1093 11447 8B6           ?A>C   A               CHANNEL # HAS TO BE =< 256
1094 1144A 70            GOYES  OUTRNG
1095 1144C 96C           ?A#0   B
1096 1144F 00            RTNYES
1097 11451 8C00  OUTRNG GOLONG =ARGERR
             00
1098                *
1099                ************************************************************
1100                ************************************************************
1101                **
1102                ** Name:(S) FIBADR  -  Find FIB entry address for a channel
1103                ** Name:(S) FIBAD-  -  Find FIB entry address for a channel
1104                **
```

```
1105                ** Category:   FILUTL
1106                **
1107                ** Purpose:  Find the FIB entry address for a given channel #
1108                **
1109                ** Entry:   A(B) = Channel #
1110                **
1111                ** EXit:  D1 & A = FIB entry address of the file
1112                **        STMTD1 = FIB entry address of the file
1113                **
1114                ** Calls: FDCH#, FFIB#
1115                **
1116                ** Used:  A,B,C,D1,R0
1117                **
1118                ** Stk lvls: +2
1119                **
1120                ********************************************************************
1121                ********************************************************************
1122                *
1123 11457 136   =FIBADR CDOEX                  SAVE D0 IN R0
1124 1145A 108          R0=C
1125 1145D AE8          B=A     B
1126 11460 7840         GOSUB   FDCH#           SEARCH THE CHNL # IN ASSIGN TABLE
1127 11464 4E0          GOC     FIBA10          CHANNEL # FOUND
1128 11467 3100 =F#NFND LC(2)   =eCHNL#
1129 1146B 6D41         GOTO    ERREX
1130 1146F 6541  BADTYP GOTO    FTYPER
1131 11473 968   FIBA10 ?A=0    B               CHANNEL # CLOSED ?
1132 11476 1F           GOYES   F#NFND          IF SO, SAY FILE # NOT FOUND
1133 11478 8E17 =FIBAD- GOSUBL  FFIB#           FIND THE FIB ENTRY ADDRESS
           E0
1134 1147E 48E          GOC     F#NFND          IF CARRY SET, NOT FOUND
1135 11481 133          AD1EX
1136 11484 1B69         D0=(5)  =STMTD1         SAVE THE FIB ENTRY ADDR IN STMTD1
           8F2
1137 1148B 140          DAT0=A  A
1138 1148E 131          D1=A
1139 11491 118          C=R0
1140 11494 521          GONC    SAVD01          (B.E.T.)
1141                *
1142                ********************************************************************
1143                ********************************************************************
1144                **
1145                ** Name:    SAVED0  -  Save D0 in RAM at STMTD0
1146                **
1147                ** Category:   GENUTL
1148                **
1149                ** Purpose:  Save D0 in system RAM "STMTD0"
1150                **
1151                ** Entry:  D0 = Value to be saved
1152                **
1153                ** Exit:  D0 saved in STMTD0
1154                **
1155                ** Calls:  None
1156                **
1157                ** Uses:  C(A), STMTD0
```

```
1158            **
1159            ** Stk lvls: 0
1160            **
1161            ****************************************************************
1162            ****************************************************************
1163            *
1164 11497 161   SAVDO+ DO=DO+ 2
1165 1149A 136  =SAVEDO CDOEX
1166 1149D 1B19 =SAVDO- DO=(5) =STMTDO
         8F2
1167 114A4 144          DATO=C A
1168 114A7 134   SAVD01 DO=C
1169 114AA 03    CH#NFD RTNCC
1170            *
1171            ****************************************************************
1172            ****************************************************************
1173            **
1174            ** Name:    FDCH#   -  Find a Channel Number in Assign Table
1175            ** Name:    FDCH#-  -  Find a Channel Number in Assign Table
1176            **
1177            ** Category:   FILUTL
1178            **
1179            ** Purpose:  Find a channel number in the Assign Table
1180            **
1181            ** Entry:
1182            **
1183            **    FDCH#:
1184            **              B(B) = Channel #
1185            **    FDCH#-:
1186            **              B(B) = Channel #
1187            **              A(A) = Assign Table buffer length
1188            **              D1 Points past buffer head
1189            **
1190            ** Exit:   Carry set => Channel # found
1191            **           D1 Points at the channel number in the Assign Table
1192            **           A(B) = 0 , If channel closed
1193            **                    = FIB # , If channel open
1194            **         Carry clear => Assign Table or channel # not found
1195            ** Calls: I/OFND
1196            ** Used:  A,C,D0,D1
1197            ** Stk lvls: +1
1198            **
1199            ****************************************************************
1200            ****************************************************************
1201            *
1202 114AC 7BF3 =FDCH#  GOSUB  BUFASN          Assign buffer
1203 114B0 59F          GONC   CH#NFD          No table means no file(error)
1204            * A(A)=Buf Len, D1 points past buffer head
1205 114B3 130  =FDCH#- DO=A
1206            *       DO=DO+ 1               Need borrow at proper time
1207 114B6 137          CD1EX                  C=D1
1208 114B9 C2           C=C+A  A               Calculate end of buffer
1209 114BB 135          D1=C                   D1 points past buffer
1210 114BE 184   FDCH40 DO=DO- 5               Decrement length
1211 114C1 48E          GOC    CH#NFD          At end of table then not found
```

```
1212 114C4 1C4          D1=D1- 5            Move pointer to next entry
1213 114C7 143          A=DAT1 A            Read entry
1214 114CA 968          ?A=0   B            Is it call level marker?
1215 114CD DD           GOYES  CH#NFD       Yes, then file not open
1216 114CF 964          ?A#B   B            Is this the right channel
1217 114D2 CE           GOYES  FDCH40        No, then loop back
1218 114D4 928          ?A=0   XS           Direct entry?
1219 114D7 91           GOYES  FDCH60        Yes, found entry
1220 114D9 7310         GOSUB  FDCH60
1221 114DD D8           B=A    A            Copy to B(B) for loop reentry
1222             * Now skip to end of call level
1223 114DF 184   FDCH50 D0=D0- 5            Decrement buffer length
1224             *      GOC    internal error which "can't happen"
1225 114E2 1C4          D1=D1- 5            Move pointer
1226 114E5 14B          A=DAT1 B
1227 114E8 96C          ?A#0   B            Is this the call marker?
1228 114EB 4F           GOYES  FDCH50       No, then loop back for next entry
1229 114ED 50D          GONC   FDCH40       (B.E.T.) Now continue search
1230             *-
1231             *-
1232 114F0 F4    FDCH60 ASR    A
1233 114F2 F4           ASR    A
1234 114F4 F4           ASR    A            A(B)=FIB#
1235 114F6 02           RTNSC
1236             *-
1237             *-
1238             *
1239             ********************************************************************
1240             ********************************************************************
1241             **
1242             ** Name:    FDFIL#  -  Find File Number in Assign Table
1243             **
1244             ** Category:   FILUTL
1245             **
1246             ** Purpose: Find a file number in the assign table
1247             **
1248             ** Entry: B(B) = File number.
1249             **
1250             ** Exit: Carry clear => File # not found
1251             **       Carry set => File # found:
1252             **        D1 @ The file number in the assign table.
1253             **
1254             ** Calls: I/OFND
1255             **
1256             ** Uses: A,C,D0,D1
1257             **
1258             ** Stk lvls: +1
1259             ********************************************************************
1260             ********************************************************************
1261             *
1262 114F8 7FA3 =FDFIL# GOSUB  BUFASN        FINS THE ASSIGN TABLE
1263 114FC 500          RTNNC              IF NOT FOUND, RETURN CARRY CLEAR
1264 114FF 130          D0=A               D0= BUFFER LENGTH
1265 11502 450          GOC    FDF#20      (B.E.T.)
1266             *
```

```
1267 11505 174   FDF#10 D1=D1+ 5
1268 11508 184   FDF#20 D0=D0- 5              DECREMENT THE BUFFER LENGTH
1269 1150B 4E9          GOC    CH#NFD         IF CARRY, REACHED END OF BUFFER
1270 1150E 143          A=DAT1 A
1271 11511 968          ?A=0   B              IS THIS A LEVEL MARK ?
1272 11514 1F           GOYES  FDF#10         IF SO, SKIP OVER IT
1273 11516 92C          ?A#0   XS             INDIRECT ENTRY ?
1274 11519 CE           GOYES  FDF#10         IF SO, SKIP OVER IT
1275 1151B 71DF         GOSUB  FDCH60
1276 1151F 964          ?A#B   B              IS THIS THE FILE # ?
1277 11522 3E           GOYES  FDF#10         IF NOT, NEXT
1278 11524 172          D1=D1+ 3              D1 @ FILE #
1279 11527 02           RTNSC                 RETURN WITH CARRY SET
1280             *
```

```
1281                    STITLE Save File Info
1282                *
1283                ****************************************************************
1284                ****************************************************************
1285                **
1286                ** Name:    SVFSPC  -  Save File Specifier.
1287                **
1288                ** Category:   FILUTL
1289                **
1290                ** Purpose:  Save file specier from FSPECx routine in STMTR0
1291                **           and STMTR1
1292                **
1293                ** Entry:  Exit condition of FSPECx
1294                **
1295                ** Exit:   STMTR0 = 1st 8 chars of file name
1296                **         STMTR1(3-0) = Last two chars of file name
1297                **         STMTR1(4) = Input of D(S)
1298                **         STMTR1(9-7) =External device address(input of D(X))
1299                **
1300                ** Calls:  None
1301                **
1302                ** Uses C,D1, STMTR0, STMTR1
1303                **
1304                ** Stk lvls:  0
1305                ****************************************************************
1306                ****************************************************************
1307                *
1308 11529 7B00   SVFSP+ GOSUB  SVFSPC
1309 1152D ACB    EXTCHK C=D    S
1310 11530 B46          C=C+1  S
1311 11533 A46          C=C+C  S
1312 11536 01           RTN
1313                *
1314 11538 1F17   SVFSPC D1=(5) =S-R0-0
           8F2
1315 1153F 1517         DAT1=A W
1316 11543 17F          D1=D1+ 16
1317 11546 118          C=R0
1318 11549 15D3         DAT1=C 4
1319 1154D 173          D1=D1+ 4
1320 11550 AFB          C=D    W
1321 11553 1554         DAT1=C S
1322 11557 172          D1=D1+ 3
1323 1155A 1553         DAT1=C X
1324 1155E 03           RTNCC
1325                *
1326                ****************************************************************
1327                ****************************************************************
1328                **
1329                ** Name:    SVFTYP  -  Save File Type Table Info
1330                **
1331                ** Category:   FILUTL
1332                **
1333                ** Purpose:
1334                **      Save file type, offset to data, create code, and copy
```

```
1335                **      code from the File Type Table entry for the given file
1336                **      type.  Info is saved in STMTR1 in coordination with
1337                **      SVFSPC.
1338                **
1339                ** Entry:
1340                **      A(A)    =  File type number (high nib = 0)
1341                **      P       =  0
1342                **
1343                ** Exit:
1344                **      P       =  0
1345                **    Carry clear:
1346                **      STMTR1(13-10)  =  File type WITHOUT privacy/security
1347                **      STMTR1(6-5)    =  Offset to data
1348                **      STMTR1(15)     =  Copy code
1349                **      STMTR1(14)     =  Create code
1350                **      C(0)           =  Create code
1351                **      D0             @  STMTR1(5)
1352                **    Carry set:
1353                **      C(3-0) =  Error code
1354                **
1355                ** Calls:      FTYPF#
1356                **
1357                ** Uses.......
1358                **  Exclusive: A(A),          C(A),D0,D1, STMTR1 exit conditions
1359                **  Inclusive: A(A),B(S),B(A),C,R0,D0,D1, STMTR1 exit conditions
1360                **
1361                ** Stk lvls:   3
1362                **
1363                ** Algorithm:
1364                **      Find File Type Table entry (error if none)
1365                **      Get "vanilla" file type (no privacy, security)
1366                **      Save away info
1367                **      Set up exit conditions
1368                **
1369                ** History:
1370                **
1371                **     Date     Programmer            Modification
1372                **    --------  ----------   ---------------------------------
1373                **    07/13/82  NZ           Added "vanilla" file type feature
1374                **    06/04/82  FH           Wrote.
1375                **
1376                ********************************************************************
1377                ********************************************************************
1378 11560 75FA =SVFTYP GOSUB  FTYPF#        Find File Type Table entry
1379 11564 503         GONC   SVFT90         Error if unrecognized type
1380 11567 134         D0=C                  Start of entry
1381 1156A 16F         D0=D0+ 16             First file type
1382 1156D 15A3        A=DAT0 4              Read "vanilla" file type to A(A)
1383 11571 1BB8        D0=(5) =S-R1-2        Write file type
           8F2
1384 11578 140         DAT0=A A               .
1385 1157B 135         D1=C                  Write create and copy code
1386 1157E 14F         C=DAT1 B               .
1387 11581 163         D0=D0+ 4               .
1388 11584 14C         DAT0=C B               .
```

```
1389 11587 172          D1=D1+ 3            Write data offset
1390 1158A 14B          A=DAT1 B             .
1391 1158D 188          D0=D0- 9             .
1392 11590 148          DAT0=A B             .
1393 11593 03           RTNCC
1394 11595 3300 SVFT90  LC(4)  =eFTYPE
           00
1395 1159B 02           RTNSC
```

```
1396                    STITLE CREATE Execute
1397            ****************************************************************
1398            ****************************************************************
1399            **
1400            ** Name:(S) CREATE  -  Statement to Create Data File
1401            **
1402            ** Category:   STEXEC
1403            **
1404            ** Purpose:
1405            **      The CREATE statement creates files of type DATA, TEXT,
1406            **      or SDATA.  The syntax is:
1407            **
1408            **         CREATE <file type> <file spec> , <size> , <# recs>
1409            **
1410            ** Entry:
1411            **      P       = 0
1412            **      D0      @ 4-nib file type in tokenized CREATE statement.
1413            **                (The file type is immediately followed by file
1414            **                specification)
1415            **
1416            ** Exit:
1417            **      P       = 0
1418            **      To NXTSTM if successful
1419            **      To BSERR  if error
1420            **
1421            ** Calls:      FSPECx,SVFSP+,SNAPSV,EXTCHK,FINDF,SNAPRS,D0=PCA,
1422            **             SVFTYP,CRTF-
1423            **
1424            ** Uses.......
1425            **  Inclusive: A-D,R0-R4,D0,D1,S11-S0,Statement and Function
1426            **             scratch RAM, SCRTCH ram, SNAPBF
1427            **
1428            ** Stk lvls:   7
1429            **
1430            ** History:
1431            **
1432            **    Date     Programmer              Modification
1433            **  --------   ----------   ----------------------------------
1434            **                SC        Designed and coded
1435            **  11/18/83      FH        Added documentation
1436            **
1437            ****************************************************************
1438            ****************************************************************
1439            *
1440            ****************************************************************
1441            ****************************************************************
1442            **
1443            ** Name:(S) pCREAT  -  Create File in External Device
1444            **
1445            ** Category:   POLL
1446            **
1447            ** Type:       POLL
1448            **
1449            **
1450            ** Purpose:
```

```
1451            **      Create a file in an external device
1452            **      This poll handles files of all copy codes except
1453            **      copy code 8.
1454            **
1455            ** Should poll be "Handled" (return with XM=0)?: Yes
1456            **
1457            ** Meaning of "Handling" Poll (what does code do if handled?):
1458            **      Create a file in an external mass memory device
1459            **
1460            ** Entry conditions for handler (registers, ST, RAM, etc.):
1461            **      B[A] = Poll number.
1462            **      HEX mode.
1463            **      P=0.
1464            **
1465            **      D(X) = Device address
1466            **      D(S) = Device type
1467            **      STMTR0 = First 8 chars of the file name
1468            **      STMTR1(3,0) = Last 2 chars of the file name
1469            **      STMTR1(6,5) = Offset to data (from file type table)
1470            **      STMTR1(9,7) = Device address
1471            **      STMTR1(13,10) = File type
1472            **      STMTR1(14) = Create code (can not be 8)
1473            **
1474            **      R2(A) = First parameter for CREATE:
1475            **      ------------------------------------------------------
1476            **      Create  Format              Meaning of this parameter
1477            **      code    Implied
1478            **      ------  ----------------    ------------------------
1479            **        0     Executable          Data length in nibs
1480            **        1     DATA(fix length)    Number of records
1481            **        2     SDATA(41C data)     Number of registers
1482            **        4     LIF1 type           File length in bytes
1483            **              (vbl len record)
1484            **
1485            **      R3(A) = Second parameter for CREATE:
1486            **      ------------------------------------------------------
1487            **       Create  Format             Meaning of this parameter
1488            **       code    Implied
1489            **      -------  ---------------    ------------------------
1490            **         0     Executable         (ignored)
1491            **         1     DATA(fix length)   Record length in bytes
1492            **         2     SDATA(41C data)    (ignored)
1493            **         4     LIF1 type(vbl len) (ignored)
1494            **
1495            **
1496            ** Normal exit conditions from handler if handled (ST, RAM,
1497            ** registers, etc.):
1498            **      Carry clear (POLL only).
1499            **      HEX mode.
1500            **      XM=0.
1501            **
1502            ** Normal exit conditions from handler if not handled (ST, RAM,
1503            ** registers, etc.):
1504            **      Carry clear (POLL only).
1505            **      HEX mode.
```

```
1506            **      XM=1.
1507            **
1508            ** Error exit conditions from handler (POLL only):
1509            **      Carry set.
1510            **      HEX mode.
1511            **      C[0-3] = Error number.
1512            **
1513            ** Available subroutine levels: 6
1514            **
1515            ** What registers/RAM may be used if handled?:
1516            **      A-D, D0, D1, P ,R0-R4, S0-S11, SCRTCH RAM
1517            **
1518            ** What registers/RAM may be used if not handled?:
1519            **      A-D, D0, D1, P
1520            **
1521            ** What registers/RAM may be used if error exit (POLL only)?:
1522            **      Anything
1523            **
1524            ** NOTE:
1525            **      No future changes to this interface should cause the
1526            **      handler to alter statement scratch!!!
1527            **
1528            ** History:
1529            **
1530            **    Date     Programmer              Modification
1531            ** --------   ----------   ----------------------------------
1532            ** 04/19/83   SC           Document
1533            **
1534            ********************************************************************
1535            ********************************************************************
1536            *
1537            ********************************************************************
1538            ********************************************************************
1539            **
1540            ** Name:(S) pCRT=8  -  Create File w/Create Code = 8
1541            **
1542            ** Category:   POLL
1543            **
1544            ** Type:       POLL
1545            **
1546            **
1547            ** Purpose:
1548            **      Create a file whose create code is 8. The file can be
1549            **      in internal memory or external mass memory device.
1550            **      The poll handler must handle all HPIL access.
1551            **
1552            ** Should poll be "Handled" (return with XM=0)?: Yes
1553            **
1554            ** Meaning of "Handling" Poll (what does code do if handled?):
1555            **      Create the file.
1556            **
1557            ** Entry conditions for handler (registers, ST, RAM, etc.):
1558            **      B[A] = Poll number.
1559            **      HEX mode.
1560            **      P=0.
```

```
1561            **
1562            **      D(X) = Device address
1563            **      D(S) = Device type
1564            **      STMTR0 = First 8 chars of the file name
1565            **      STMTR1(3,0) = Last 2 chars of the file name
1566            **      STMTR1(6,5) = Offset to data (from file type table)
1567            **      STMTR1(9,7) = Device address
1568            **      STMTR1(13,10) = File type
1569            **      STMTR1(14) = Create code (can not be 8)
1570            **
1571            **      R2(A) = First parameter for CREATE:
1572            **      ------------------------------------------------------
1573            **      Create  Format              Meaning of this parameter
1574            **      code    Implied
1575            **      ------  ----------------    ------------------------
1576            **        0     Executable          Data length in nibs
1577            **        1     DATA(fix length)    Number of records
1578            **        2     SDATA(41C data)     Number of registers
1579            **        4     LIF1 type           File length in bytes
1580            **              (vbl len record)
1581            **
1582            **      R3(A) = Second parameter for CREATE:
1583            **      ------------------------------------------------------
1584            **       Create  Format             Meaning of this parameter
1585            **       code    Implied
1586            **      -------  ---------------    ------------------------
1587            **         0     Executable         (ignored)
1588            **         1     DATA(fix length)   Record length in bytes
1589            **         2     SDATA(41C data)    (ignored)
1590            **         4     LIF1 type(vbl len) (ignored)
1591            **
1592            **
1593            ** Normal exit conditions from handler if handled (ST, RAM,
1594            ** registers, etc.):
1595            **      Carry clear
1596            **      HEX mode.
1597            **      XM=0.
1598            **
1599            ** Normal exit conditions from handler if not handled (ST, RAM,
1600            ** registers, etc.):
1601            **      Carry clear
1602            **      HEX mode.
1603            **      XM=1.
1604            **
1605            ** Error exit conditions from handler (POLL only):
1606            **      Carry set.
1607            **      HEX mode.
1608            **      C[0-3] = Error number.
1609            **
1610            ** Available subroutine levels: 6
1611            **
1612            ** What registers/RAM may be used if handled?:
1613            **      A-D, D0, D1, P ,R0-R4, ST, SCRTCH RAM
1614            **
1615            ** What registers/RAM may be used if not handled?:
```

```
1616                **      A-D, D0, D1, P
1617                **
1618                ** What registers/RAM may be used if error exit (POLL only)?:
1619                **      Anything
1620                **
1621                ** NOTE:
1622                **      No future changes to this interface should cause the
1623                **      poll handler to alter Statement Scratch!
1624                **
1625                ** History:
1626                **
1627                **    Date     Programmer             Modification
1628                **  --------   ----------   ----------------------------------
1629                **  04/19/83   SC           Document
1630                **
1631                ********************************************************************
1632                ********************************************************************
1633 1159D 00000        REL(5) =CREADC
           0
1634 115A2 00000        REL(5) =CREATP
           0
1635 115A7 163  =CREATE D0=D0+ 4              JUMP OVER THE FILE TYPE TOKEN
1636 115AA 7232         GOSUB  fspecx         PROCESS THE FILE NAME
1637 115AE 501          GONC   CTE100
1638                *
1639 115B1 68A0         GOTO   CRTERR
1640                *
1641 115B5 3100 =FTYPER LC(2)  =eFTYPE
1642                *
1643 115B9 8C00  ERREX  GOLONG =MfErr
           00
1644                *
1645 115BF 766F  CTE100 GOSUB  SVFSP+         SAVE THE FILE SPECIFIER FIRST
1646                *
1647 115C3 777B         GOSUB  GETARG         PROCESS THE ARGUMENTS
1648                *
1649                * Now check if the file exist yet
1650                *
1651 115C7 112          A=R2
1652 115CA 11B          C=R3
1653 115CD AF7          D=C    W
1654 115D0 8F00         GOSBVL =SNAPSV        SAVE R2 & R3
           000
1655                *
1656 115D7 1B17         D0=(5) =STMTRO
           8F2
1657 115DE 1527         A=DAT0 W              A= FILE NAME
1658 115E2 16F          D0=D0+ 16
1659 115E5 163          D0=D0+ 4
1660 115E8 1564         C=DAT0 S
1661 115EC AC7          D=C    S              D(S) = DEVICE TYPE
1662 115EF 162          D0=D0+ 3
1663 115F2 1563         C=DAT0 X
1664 115F6 AB7          D=C    X              D(X) = DEVICE ADDRESS
1665 115F9 703F         GOSUB  EXTCHK         CHECK IF IS AN EXTERNAL DEVICE
```

```
1666 115FD 4F0          GOC    CTE110          IF SO, NO NEED TO CHECK DUPL. FILE
1667               *
1668 11600 7E25         GOSUB  findf
1669 11604 480          GOC    CTE110          FILE NOT FOUND IS OK
1670               *
1671 11607 8C00         GOLONG =RNMERR         IF FOUND, SAY FILE EXIST
           00
1672               *
1673 1160D 8E00  CTE110 GOSUBL =Snaprs         RESTORE R2 & R3
           00
1674 11613 102          R2=A
1675 11616 AFB          C=D    W
1676 11619 10B          R3=C
1677               *
1678 1161C 8E00         GOSUBL =DO=PCA         NOW GO BACK TO SEE THE TYPE
           00
1679 11622 167          DO=DO+ 8               DO POINTS AT FILE TYPE TOKEN
1680 11625 D0           A=0    A
1681 11627 15A3         A=DATO 4
1682 1162B 713F         GOSUB  SVFTYP
1683 1162F 498          GOC    ERREX
1684 11632 D5           B=C    A
1685 11634 90D          ?B#0   P               IS IT A MAINFRAME FILE ?
1686 11637 60           GOYES  CTE120
1687               *
1688 11639 6B7F         GOTO   FTYPER          IF SO, FILE TYPE ERROR
1689               *
1690 1163D 1B58  CTE120 DO=(5) (=STMTR1)+4
           8F2
1691 11644 1564         C=DATO S
1692 11648 AC7          D=C    S               D(S) = DEVICE TYPE
1693 1164B 162          DO=DO+ 3
1694 1164E 146          C=DATO A
1695 11651 D7           D=C    A               D(B) = PORT NUMBER
1696 11653 7A70         GOSUB  CRTF-
1697 11657 560          GONC   CTE300          If no error, done
1698               *
1699 1165A 62B4  CRTERR GOTO   OPNERR          UNRECOGNIZED FILE TYPE
1700               *
1701 1165E 8C00  CTE300 GOLONG =NEXTst         To NXTSTM
           00
1702               *
1703               *
1704               ****************************************************************
1705               ****************************************************************
1706               **
1707               ** Name:    CRFSUB  -  Create a File in Mainframe
1708               ** Name:(S) CRFSB-  -  Create a File in Mainframe
1709               **
1710               ** Category:   FILUTL
1711               **
1712               ** Purpose: Create a file in mainframe
1713               **
1714               ** ENTRY:
1715               **           P      = 0
```

```
1716                **          STMTR0 = FILE NAME
1717                **          STMTR1(4) = DEVICE TYPE
1718                **          STMTR1(8-7) = PORT #
1719                **          STMTR1(15) = FILE COPY CODE FROM FILE TYPE TABLE
1720                **          S-R1-0(13-10) = FILE TYPE
1721                **          R1  =  ADDRESS OF FILE HEADER ALREADY CREATED BY
1722                **                 CREATF.  FILE NAME, COPY CODE, AND FILE
1723                **                 TYPE WILL BE FILLED IN.
1724                **
1725                ** EXIT:  FILE HEADER ALL BEEN PROPERLY FILLED
1726                **          A     =  FILE CHAIN LENGTH
1727                **          C(A)  =  0
1728                **          D1    @  PAST THE FILE CHAIN LENGTH FIELD
1729                **          D0    @  S-R1-3 (COPY CODE)
1730                **          R1    =  ADDRESS OF FILE HEADER
1731                **          P     =  0
1732                **
1733                ** Calls: CREATF, A-MULT
1734                **
1735                ** USES:
1736                **        Inclusive: A(A),C,D0,D1
1737                **
1738                ** Stk lvls: 0
1739                **
1740                ********************************************************************
1741                ********************************************************************
1742                *
1743 11664 119   =CRFSB- C=R1
1744 11667 135          D1=C                    D1 POINTS AT START OF HEADER
1745 1166A 1B17         D0=(5) =S-R0-0
           8F2
1746 11671 1567         C=DAT0 W                WRITE FILE NAME
1747 11675 1557         DAT1=C W
1748 11679 16F          D0=D0+ 16
1749 1167C 169          D0=D0+ 10               D0 POINTS AT SAVED FILE TYPE
1750 1167F 17F          D1=D1+ 16               D1 PTS AT FILE TYPE IN FILE HEADER
1751 11682 146          C=DAT0 A
1752 11685 15D3         DAT1=C 4                WRITE FILE TYPE
1753 11689 174          D1=D1+ 5                D1 PTS AT L. S. NIB OF FLAGS
1754 1168C 164          D0=D0+ 5                D0 PTS AT SAVED COPY CODE
1755 1168F 14E          C=DAT0 B
1756 11692 15D0         DAT1=C 1                WRITE COPY CODE
1757 11696 17A          D1=D1+ 11               D1 POINTS FILE CHAIN LENGTH
1758 11699 143          A=DAT1 A
1759 1169C 174          D1=D1+ 5                D1 PTS PASSED FILE CHAIN LENGTH
1760 1169F D2           C=0    A
1761 116A1 03           RTNCC
1762                *
1763                ************************
1764                *
1765 116A3 1F17  POLLFF D1=(5) =STMTR0
           8F2
1766 116AA 1537         A=DAT1 W
1767 116AE 100          R0=A
1768 116B1 17F          D1=D1+ 16
```

```
1769 116B4 1537          A=DAT1 W
1770 116B8 101           R1=A
1771 116BB 8C00  poll    GOLONG =POLL
          00
1772               *
1773               *
1774               ********************************************************************
1775               ********************************************************************
1776               **
1777               ** Name:(S) CRTF    -  Create File in MAIN, PORT, or HPIL
1778               **
1779               ** Category:   FILUTL
1780               **
1781               ** Purpose:
1782               **      Create a file of arbitrary type in memory or on an
1783               **      external device.
1784               **
1785               ** Entry:
1786               **      A      =  First 8 chars of file name
1787               **      D(S)   =  FIB device code
1788               **      D(A)   =  FIB device address:
1789               **                D(B)  =  Port# and Extender# for PORT
1790               **                D(X)  =  Device address for HPIL device
1791               **      P      =  0
1792               **      R0     =  Last two chars of file name if HPIL device
1793               **      R1(A)  =  File type (high nib = 0)
1794               **      R2(A)  =  First parameter for create:
1795               **                ----------------------------------------
1796               **                Create   Format    Meaning of This
1797               **                 Code    Implied      Parameter
1798               **                ------   -------   --------------------
1799               **                  0      Standard  Data length in nibs
1800               **                  1      DATA      Number of records (can
1801               **                                     be 0 if not HPIL)
1802               **                  2      SDATA     Number of records (can
1803               **                                     be 0 if not HPIL)
1804               **                  4      Vbl Rec   Number of bytes in file
1805               **                  8      OEM       Unknown; poll for len
1806               **
1807               **      R2(9:5)=  Address of data in RAM/ROM to copy to the
1808               **                newly created file (none if zero)
1809               **
1810               **      R3(A)  =  Second parameter for create:
1811               **                ----------------------------------------
1812               **                Create   Format    Meaning of This
1813               **                 Code    Implied      Parameter
1814               **                ------   -------   --------------------
1815               **                  0      Standard  (Ignored)
1816               **                  1      DATA      Record length in bytes
1817               **                                     (256 default)
1818               **                  2      SDATA     (Ignored; set to 8)
1819               **                  4      Vbl Rec   (Ignored)
1820               **                  8      OEM       Unknown; poll for len
1821               **
1822               ** Exit:
```

```
1823          **      P      =  0
1824          **      R2(A)  =  File length in nibbles (chain length)
1825          **      R3(A)  =  Entry state (updated if default condition)
1826          **     Carry set:
1827          **      C(A)   =  Error code:
1828          **                  "Not Implemented"
1829          **     Carry clear:
1830          **      D(S)   =  File device code
1831          **       (X)   =  Device address
1832          **      R1     =  Address of file header if file in memory
1833          **      D1     @  Start of data if file is in memory
1834          **
1835          ** Calls:      SVFPSC,SVFTYP,POLL,A-MULT,CRETF+,CRFSB-,INITMF
1836          **
1837          ** Uses.......
1838          **  Exclusive: A,B(S,A),C,D(S),D0,D1,   R1,R2
1839          **  Inclusive: A-D,            D0,D1,R0-R4,STMTR0,STMTR1,
1840          **             SCRTCH,S11-S0
1841          **
1842          ** Stk lvls:   6 - If file created on plug-in, else 5
1843          **
1844          ** NOTE:
1845          **      This routine can only create BASIC, TEXT, and 41C data
1846          **      in memory at the moment.
1847          **
1848          ** Algorithm:
1849          **      Save away file spec and file type info
1850          **      Compute data length from parameters
1851          **      Compute and add on subheader length
1852          **      If device is not MAIN then
1853          **         Error exit for now (not implemented)
1854          **      Create file header in memory
1855          **      Fill in name, etc.
1856          **      Initialize file according to create code
1857          **
1858          ** History:
1859          **
1860          **    Date     Programmer            Modification
1861          **   --------  ----------   ---------------------------------
1862          **   09/24/82  SC           Add code to create and initialize
1863          **                          file in HP-IL device.
1864          **                          POLL to create OEM file and the
1865          **                          POLL handler has to do it all.
1866          **   07/21/82  NZ           Modified entry condition locations
1867          **   07/13/82  NZ           Added field (R2(9:5)) for address
1868          **                          of data to copy to the file after
1869          **                          creation; modified exit code to
1870          **                          use D0 instead of D1 to get stated
1871          **                          exit conditions; changed R2 exit
1872          **                          conditions
1873          **   06/01/82  FH           Wrote from looking at code for
1874          **                          CREATE execute and CRBAS (create
1875          **                          BASIC).  Needed for TRANSFORM.
1876          **
1877          ************************************************************
```

```
1878                ****************************************************************
1879 116C1 737E =CRTF   GOSUB SVFSPC          Save file info for create
1880 116C5 111          A=R1                  Fetch and save file type info
1881 116C8 749E         GOSUB  SVFTYP          .
1882 116CC 400          RTNC                   . and return if error
1883 116CF D5           B=C    A              Save create code
1884                *
1885                *  Compute data length from create parameters if needed
1886                *
1887 116D1 112   CRTF-  A=R2                   Recall data len
1888 116D4 A05          B=B+B  P               Not Create Code 8 (Poll)?
1889 116D7 5C0          GONC   CRTF05            GOYES
1890 116DA 7DDF         GOSUB  poll            Poll for Create Code 8 (has to do
1891 116DE 32           CON(2) =pCRT=8          . it all and initialize it)
1892 116E0 6C60         GOTO   CRTF45
1893                *
1894 116E4 909   CRTF05  ?B=0  P               CREATE CODE =0 ?
1895 116E7 45           GOYES  CRTF40          IF SO , GOTO CRTF40
1896 116E9 A05          B=B+B  P               LIF1 format?
1897 116EC 4B3          GOC    CRTF30            GOYES
1898                *
1899                * The file type is either DATA or SDATA,
1900                * no argument should be > 65,535
1901                *
1902 116EF 11B          C=R3                   Fetch record length
1903 116F2 24           P=     4
1904 116F4 90C          ?A#0   P
1905 116F7 70           GOYES  CRTF10
1906 116F9 90E          ?C#0   P
1907 116FC 20           GOYES  CRTF10
1908 116FE 20    CRTF10 P=     0
1909 11700 4F1          GOC    CRTFer          Carry set if argument > 65,535
1910                *
1911 11703 A05          B=B+B  P               Not 41C data file format?
1912 11706 570          GONC   CRTF20            GOYES
1913 11709 D2           C=0    A               Set rec len to 8 for 41C file
1914 1170B 308          LC(1)  8
1915 1170E 8AE   CRTF20 ?C#0   A               Rec len specified?
1916 11711 50           GOYES  CRTF25
1917 11713 B26          C=C+1  XS              Default is 256 (for DATA)
1918 11716 10B   CRTF25 R3=C                   .
1919 11719 79C0         GOSUB  a-mult          Compute len in bytes
1920 1171D 481          GOC    CRTF35          Branch if no overflow
1921                *
1922 11720 3300  CRTFer LC(4)  =eMEM           Error: "Not enough mem"
           00
1923 11726 02           RTNSC
1924                *
1925                * Round the file size to even number of bytes for test file
1926                *
1927 11728 E4    CRTF30 A=A+1  A
1928 1172A 45F          GOC    CRTFer
1929 1172D 81C          ASRB
1930 11730 A74          A=A+A  W
1931 11733 102          R2=A                   Update the size in R2 too
```

```
1932                *
1933 11736 C4     CRTF35 A=A+A  A              Compute len in nibs
1934 11738 47E            GOC    CRTFer
1935                *
1936 1173B ACB    CRTF40 C=D    S
1937 1173E B46           C=C+1  S              TURN F INTO 0
1938 11741 A46           C=C+C  S
1939 11744 581           GONC   CRTF50         NOT TO CREATE IN HP-IL DEVICE
1940 11747 707F          GOSUB  poll
1941 1174B 90            CON(2) =pCREAT
1942 1174D 400    CRTF45 RTNC                  RETURN WITH CARRY SET IF ERROR
1943 11750 831           ?XM=0                 Handled?
1944 11753 60            GOYES  CRTF46
1945 11755 6F3E          GOTO   SVFT90         Error: "Illegal file type"
1946 11759 6070   CRTF46 GOTO   CRTF70
1947                *-
1948                *-
1949                *
1950                *  Add on subheader and file header length
1951                *
1952 1175D 11B    =CRTF50 C=R3
1953 11760 D5            B=C    A              B(A) = Record length
1954 11762 1F68          D1=(5) (=S-R1-0)+5    Fetch offset to data
           8F2
1955 11769 D2            C=0    A               .
1956 1176B 14F           C=DAT1 B               .
1957 1176E CA            A=A+C  A              Add it in
1958                *
1959                * Write updated (ie real) file length in nibbles in R2(A)
1960                *
1961 11770 11A           C=R2                  Preserve upper nibs of R2!
1962 11773 D6            C=A    A
1963 11775 8E00          GOSUBL =Cslc5
           00
1964 1177B D9            C=B    A              Save record length in R2
1965 1177D 10A           R2=C
1966                *
1967 11780 D2            C=0    A              Clear high nibbles
1968 11782 3102          LC(2)  =oFLENh        Add in rest of file header len
1969 11786 C2            C=C+A  A               .
1970                *
1971                *  Now create the file
1972                *
1973                *-
1974 11788 8F00          GOSBVL =CRETF+        Create mainframe file
           000
1975 1178F 400           RTNC
1976 11792 11A           C=R2                  Restore record length to R3
1977 11795 10B           R3=C
1978 11798 8E00          GOSUBL =csrc5         Restore R2
           00
1979 1179E 10A           R2=C
1980                *
1981                *  Initialize the file header
1982                *
```

```
1983 117A1 7FBE          GOSUB  CRFSB-          Write name, type, codes
1984 117A5 180           D0=D0- 1               Fetch create code into B
1985             *       C=DAT0 P                .
1986             *       B=C    A
1987 117A8 1564          C=DAT0 S                .
1988 117AC AC5           B=C    S               B(S) is create code
1989 117AF 184           D0=D0- 5               Fetch file type into A
1990 117B2 146           C=DAT0 A                .
1991 117B5 F6            CSR    A                .
1992 117B7 D5            B=C    A               B(A) is file type
1993             *
1994             * B(A) is file type, B(S) is create code, A(A) is file chain len
1995             *
1996 117B9 183           D0=D0- 4               Fetch offset to data
1997 117BC D2            C=0    A
1998 117BE 14E           C=DAT0 B               C(A) is offset to data of file
1999 117C1 EA            A=A-C  A               A(A) is data length of file
2000             *
2001             *  Now initialize the subheader and data portions
2002             *
2003 117C3 7620          GOSUB  INITMF
2004 117C7 400           RTNC
2005             *
2006             *  Set up successful return condition
2007             *
2008 117CA 1F58 CRTF70   D1=(5) (=S-R1-0)+4     Fetch device codes
          8F2
2009 117D1 1574          C=DAT1 S                . (FIB device code)
2010 117D5 172           D1=D1+ 3                .
2011 117D8 147           C=DAT1 A                . (device address
2012 117DB AF7           D=C    W                .
2013 117DE 03            RTNCC
2014             *
2015             *
2016 117E0 8C00 =fspecx GOLONG =FSPECx
          00
2017 117E6 8D00 =a-mult GOVLNG =A-MULT
          000
2018             *
2019             ********************************************************************
2020             ********************************************************************
2021             **
2022             ** Name:    INITMF  -  Initialize Memory File After Create
2023             **
2024             ** Category:   FILUTL
2025             **
2026             ** Purpose:
2027             **      Initialize subheader and data portion of a file after
2028             **      creation.  Works only for memory files at the moment.
2029             **
2030             ** Entry:
2031             **      A(A)   =  Data length of file in nibs
2032             **      B(A)   =  File type # (high nib = 0)
2033             **       (S)   =  Create code for file
2034             **      C(A)   =  Offset to data for file type
```

```
2035            **      D1      @  Past file chain length field in file header
2036            **      R1      =  Start of file header
2037            **      R3      =  Length in bytes of logical record (fixed data)
2038            **      P       =  0
2039            **
2040            ** Exit:
2041            **      D1      @  Start of data portion of file (at next header
2042            **                    in chain if no data portion)
2043            **      R1      =  Entry state
2044            **      Carry clear
2045            **
2046            ** Calls:       STUFF-
2047            **
2048            ** Uses.......
2049            **  Exclusive: A,B(A),C(A),D0,D1,R0
2050            **  Inclusive: A,B(A),C(A),D0,D1,R0
2051            **
2052            ** Stk lvls:   1
2053            **
2054            ** Algorithm:
2055            **      Zero out subheader
2056            **      Case of CREATE CODE:
2057            **         0 (Standard):       If BASIC then
2058            **                                Write EOL to subheader
2059            **
2060            **         1 (DATA):           Write # of records and record
2061            **                                  length to Impl Field
2062            **                             Stuff data portion with -1
2063            **
2064            **         2 (41C Data):       Stuff data portion with 0
2065            **
2066            **         4 (LIF1 format):    Stuff data portion with -1
2067            **
2068            **         8 (Alien):          Zero subheader and return
2069            **      Endcase
2070            **      Set up exit condition
2071            **
2072            ** History:
2073            **
2074            **    Date      Programmer            Modification
2075            **  --------    ----------   ---------------------------------*
2076            **  07/22/82    NZ           Modified entry conditions, type 1
2077            **                           Modified initialization of type 1
2078            **  07/21/82    NZ           Added file type 8 handling
2079            **  07/13/82    NZ           Fixed bug in fBASIC test:A#B=>C#B
2080            **  06/08/82    FH           Designed and coded.
2081            **
2082            ********************************************************************
2083            ********************************************************************
2084            *
2085            *  Clear subheader
2086            *
2087 117ED 100  =INITMF R0=A                  Save away data length
2088 117F0 137          CD1EX                Subhdr len = offset-length field
2089 117F3 1C4          D1=D1- =1FLENh        .
```

```
2090 117F6 134          DO=C                Set DO @ Imp Fld while handy
2091 117F9 137          CD1EX               Set C to subheader length
2092              *
2093              * C[A] is subheader length, DO @ Implementation field
2094              *
2095 117FC 8F00         GOSBVL =WIPOUT      Clear subheader
           000
2096 11803 94D          ?B#0   S            Not Standard file format?
2097 11806 D1           GOYES  INIT20
2098              *
2099              *  Create Code 0:   Standard File Format
2100              *
2101              *    (data portion not initialized;
2102              *     only BASIC subheader gets special treatment)
2103              *
2104 11808 3441         LC(5)  =fBASIC
           2E0
2105 1180F 8A5          ?C#B   A            Not create BASIC file?
2106 11812 F0           GOYES  INIT10
2107 11814 3100         LC(2)  =tEOL        Write EOL to end of subheader
2108 11818 1C1          D1=D1- 2             .
2109 1181B 14D          DAT1=C B             .
2110 1181E 171          D1=D1+ 2            Restore output condition
2111 11821 03   INIT10  RTNCC
2112              *-
2113              *-
2114 11823 A45  INIT20  B=B+B  S             Alien format?
2115 11826 4AF          GOC    INIT10         GOYES to exit
2116 11829 137          CD1EX               Save away data start pointer
2117 1182C 135          D1=C                 .
2118 1182F 128          CR0EX                .  and recall data len
2119 11832 A45          B=B+B  S            Variable length format?
2120 11835 444          GOC    INIT70         GOYES
2121 11838 AF0          A=0    W
2122 1183B A45          B=B+B  S            41C Register data format?
2123 1183E 414          GOC    INIT75
2124              *
2125              *  Create Code 1:   DATA File Format
2126              *
2127              *    (Write out Param1 and Param2 to Implementation Field;
2128              *     initialize records)
2129              *
2130              * C[A] is NOW data length
2131              *
2132 11841 DA    INIT30 A=C    A
2133 11843 81C          ASRB                Length of data in bytes
2134 11846 100          R0=A                Store length for now
2135 11849 113          A=R3                Number of records
2136 1184C 163          D0=D0+ 4            Second parameter=length in bytes
2137 1184F 1583         DAT0=A 4            Write len of records
2138 11853 183          D0=D0- 4            Back to # of records
2139 11856 23           P=     3
2140 11858 AF2          C=0    W            High nibs are clear from above
2141 1185B A96          C=A    WP           C is now record length in bytes
2142 1185E 110          A=R0                A is file length in bytes
```

```
2143                *
2144                * Divide A by C to find number of records
2145                *
2146 11861 8E00        GOSUBL =IDIV          Returns Quotient in A, Rem in B
          00
2147 11867 20          P=     0
2148 11869 1583        DATO=A 4              Write out the result!
2149                *
2150                * Now initialize the data area (D1 is @ data start)
2151                *
2152 1186D 133         AD1EX
2153 11870 131         D1=A                  Get data start to A[A]
2154 11873 120         AROEX                 Data start to R0, length to A[A]
2155 11876 C4          A=A+A  A              Convert back to nibs
2156 11878 D6          C=A    A
2157                *-
2158                *
2159                *  Create Code 2:   41C Data File Format
2160                *
2161                *  Create Code 4:   LIF1 (Variable Length) Format
2162                *
2163 1187A AF0  INIT70  A=0    W
2164 1187D A7C         A=A-1  W              Fill with -1
2165 11880 8E00 INIT75  GOSUBL =stuff        Fill with -1 or 0
          00
2166                *-
2167 11886 118         C=R0                  Restore data start pointer
2168 11889 135         D1=C                   .
2169 1188C 03          RTNCC
2170                *
2171                ************************************************************************
2172                ************************************************************************
2173                **
2174                ** Name:(S) IOFSCR  -  I/O Find for Available Scratch Buffer
2175                **
2176                ** Category:   BUFUTL
2177                **
2178                ** Purpose:
2179                **      Returns available scratch buffer ID
2180                **
2181                ** Entry:
2182                **      P      =  0
2183                **
2184                ** Exit:
2185                **      P      =  0
2186                **      Carry clear => Available Buffer ID in C(X)
2187                **                set => No available scratch buffers
2188                **                       C(X)=000
2189                **
2190                ** Calls:      I/OFND
2191                **
2192                ** Uses.......
2193                **      A, C(A), D1
2194                **
2195                ** Stk lvls:   1
```

```
2196                **
2197                **
2198                ** Detail:
2199                **      Scratch buffer ID's range from E00 (bSCRTC) to FFF
2200                **
2201                ** History:
2202                **
2203                **    Date      Programmer          Modification
2204                ** --------   ----------   ---------------------------------
2205                ** 02/08/83   S.W.         Added documentation
2206                **
2207                ********************************************************************
2208                ********************************************************************
2209                *
2210 1188E 3200 =IOFSCR LC(3)  =bSCRTC
           E
2211 11893 7320  IOFSC5 GOSUB  I/OFND
2212 11897 500          RTNNC                  Buffer not found?
2213 1189A B36          C=C+1  X               Try next ID#
2214 1189D 55F          GONC   IOFSC5          Not yet reached FFF ?
2215 118A0 01           RTN                    No unused scratch buffer ID's
2216                *
```

```
2217                    STITLE I/O BUFFER UTILITY ROUTINES
2218                ****************************************************************
2219                ****************************************************************
2220                **
2221                ** Name:(S) I/OFND  -  I/O Buffer Find
2222                ** Name:(S) IOFNDO  -  I/O Buffer Find
2223                **
2224                ** Category:   BUFUTL
2225                **
2226                ** Purpose:    Find the specified I/O buffer
2227                **
2228                **             IOFNDO looks for the buffer ID specified in C(X).
2229                **
2230                **             I/OFND sets the high bit of the buffer ID
2231                **             specified in C(X), then looks for that buffer.
2232                **             (Buffer IDs with the high bit clear are those
2233                **             which will be deallocated at the next configura-
2234                **             tion).
2235                **
2236                ** Entry:    C(X)= Buffer ID#
2237                **
2238                ** Exit:   C(X)= Buffer ID#
2239                **           Carry set=> Match found
2240                **                       D1   points past buffer header
2241                **                       A(A) Buffer length field
2242                **                       C(S)=#addresses to update in buffer
2243                **           Carry clr=> No match
2244                **
2245                ** Calls:    none
2246                **
2247                ** Uses:     A, C(A), C(S), D1
2248                **
2249                ** Stack lvls: 0
2250                **
2251                ** Detail:   Buffer length field in header reflects the amount
2252                **           of available scratch space in that buffer, but is
2253                **           not the entire length of the buffer (eg doesn't
2254                **           include 7 nibbles for the header)
2255                **
2256                ** History:
2257                **
2258                **    Date     Programmer   Modifications
2259                **  --------   ----------   -------------
2260                **  07/04/82   S.W.         Added documentation
2261                **  02/10/83   S.W.         Added 1 nibble to header front
2262                **  03/10/83   S.W.         Save Leeway setting in B(S)
2263                **  03/14/83   M.B.         Packed 3 nibs in I/OFN+
2264                **
2265                ****************************************************************
2266                ****************************************************************
2267                *
2268 118A2 3230 =BUFFIB LC(3)  =bFIB         FIND FIB BUFFER
               8
2269 118A7 6210         GOTO   I/OFND
2270 118AB 3240 =BUFASN LC(3)  =bASSGN      FIND ASSIGN BUFFER
```

```
          8
2271 118B0 6900          GOTO   I/OFND
2272              *
2273 118B4 80FE  I/OFN+  CPEX   14              C(14) for leeway check.
2274 118B8 20            P=     0
2275              *
2276 118BA 0B    =I/OFND CSTEX
2277 118BC 85B           ST=1   11
2278 118BF 0B            CSTEX
2279 118C1 1F17 =IOFNDO  D1=(5) =MAINEN         START OF IO BUFFER AREA
          5F2
2280 118C8 143           A=DAT1 A
2281 118CB 131           D1=A
2282 118CE 6B10          GOTO   IOFND5          GOTO IOFND5
2283 118D2 B3A   IOFND3  A=A-C  X
2284 118D5 BB8           A=-A   X               Clears carry only if A(X)=0.
2285 118D8 BF4           ASR    W               SHIFT IN LENGTH
2286 118DB F4            ASR    A
2287 118DD F4            ASR    A
2288 118DF 5E3           GONC   RTNSCj          BUFFER FOUND!
2289 118E2 137           CD1EX
2290 118E5 C2            C=A+C  A               POINT TO NEXT BUFFER
2291 118E7 137           CD1EX
2292 118EA 1574  IOFND5  C=DAT1 S               Read in #addresses
2293 118EE 170           D1=D1+ 1               Step over, pt to ID
2294 118F1 15B5          A=DAT1 6               READ HEADER
2295 118F5 175           D1=D1+ 6               POINT PAST HEADER
2296 118F8 93C           ?A#0   X               NOT AT END OF BUFFER AREA?
2297 118FB 7D            GOYES  IOFND3          IF NOT, REPEAT IOFND3
2298 118FD 03            RTNCC                  RETURN CARRY CLEAR
2299              *
2300              ************************************************************
2301              ************************************************************
2302              **
2303              ** Name:(S) I/ORES  -  I/O Buffer Restore
2304              **
2305              ** Category:   BUFUTL
2306              **
2307              ** Purpose:  Sets high bit of buffer ID to preserve buffer
2308              **
2309              ** Entry:    C(X) IS BUF ID#
2310              **
2311              ** Exit:     CARRY SET=> BUFFER FOUND AND HIGH BIT OF ID# SET.
2312              **                       D1 POINTS PAST HEADER.
2313              **                       C(X) IS ID# WITH HIGH BIT SET.
2314              **
2315              **           CARRY CLR=> BUFFER NOT FOUND.
2316              **
2317              ** Calls:    I/OFND
2318              **
2319              ** Uses:     A, C, D1
2320              **
2321              ** Stack lvls: 1
2322              **
2323              ** History:
```

```
2324            **
2325            **    Date     Programmer   Modifications
2326            **  --------   ----------   -------------
2327            **  07/04/82   S.W.         Added documentation
2328            **
2329            ****************************************************************
2330            ****************************************************************
2331            *
2332            *
2333 118FF 0B   =I/ORES CSTEX
2334 11901 84B          ST=0   11             EXPLICITLY CLR HIGH BIT
2335 11904 0B           CSTEX
2336 11906 77BF         GOSUB  IOFNDO
2337 1190A 5FA          GONC   I/OFND         MAYBE HIGH BIT ALREADY SET
2338            * SET HIGH BIT
2339 1190D 1C3          D1=D1- 4              POINT BACK TO ID HINIB
2340 11910 0B           CSTEX
2341 11912 85B          ST=1   11
2342 11915 0B           CSTEX
2343 11917 1552         DAT1=C XS             SET HIGH BIT IN HEADER ID
2344 1191B 173          D1=D1+ 4              POINT PAST HEADER
2345 1191E 02    RTNSCj RTNSC
2346            *
2347            ****************************************************************
2348            ****************************************************************
2349            **
2350            ** Name:(S) I/OCON  -  I/O Buffer Contract From Buffer End
2351            ** Name:    IOCNDO  -  I/O Buffer Contract From Buffer Middle
2352            **
2353            ** Category:   BUFUTL
2354            **
2355            ** Purpose:
2356            **      Contract an I/O buffer.
2357            **
2358            **      I/OCON contracts the buffer from its end, losing data
2359            **      stored at the end of the buffer.
2360            **
2361            **      IOCNDO contracts a specified section of the buffer.
2362            **
2363            ** Entry:
2364            **      C(X)   =  Buffer number
2365            **      B(A)   =  Amount to shrink existing buffer
2366            **                 A positive number - not to exceed 00FFF
2367            **      2 entry points:
2368            **            1) I/OCON - No additional requirements
2369            **            2) IOCNDO - D0 points to the beginning of the
2370            **               block that is to be deleted.
2371            **
2372            ** Exit:
2373            **      Carry clear=> Buffer not found
2374            **               set=> Buffer contracted specified amount
2375            **                     D1 points past buffer header
2376            **                     D0 points 1 nibble past front of header
2377            **                       (at buffer ID)
2378            **                     P=0
```

```
2379                **
2380                ** Calls:      I/OFND, IDLNSV, MOVEMU, PTRADJ
2381                **
2382                ** Uses.......
2383                **  Exclusive: A-D, D0, D1
2384                **  Inclusive: A-D, D0, D1
2385                **
2386                ** Stk lvls:   3
2387                **
2388                ** Detail:
2389                **      If amount to contract given in B(A) is greater than
2390                **      the current buffer size, the buffer is collapsed.
2391                **      See I/OCOL
2392                **
2393                ** History:
2394                **
2395                **    Date      Programmer            Modification
2396                **  --------   ----------   ---------------------------------
2397                **  07/04/82   S.W.         Added documentation
2398                **  09/13/83   S.W.         Modified doc. to show stk lvls=3
2399                **
2400                ********************************************************************
2401                ********************************************************************
2402                *
2403                *
2404 11920 D0     =I/OCON A=0     A
2405 11922 130          D0=A                    FLAG DEFAULT SHRINK
2406 11925 719F =IOCNDO GOSUB   I/OFND
2407 11929 500          RTNNC
2408 1192C DD           CBEX    A               C(A)=AMT TO CON;B(X)=BUF ID
2409 1192E B3E          C=A-C   X               DESIRED BUFFER LENGTH
2410 11931 454          GOC     IONULL
2411 11934 72F0         GOSUB   IDLNSV          SAVE BUF LEN & ID IN D
2412 11938 E8           B=B-A   A               PTR ADJUST OFFSET
2413 1193A 137          CD1EX
2414 1193D 06           RSTK=C                  SAVE BUFST
2415 1193F 132          ADOEX                   D0=BUF LENGTH; A(A)=BGN DEST.
2416 11942 131          D1=A
2417 11945 8AC          ?A#0    A               BEGIN DEST SPECIFIED?
2418 11948 11           GOYES   IOCN10
2419 1194A 132          ADOEX                   A(A)=BUF LENGTH
2420 1194D CA           A=A+C   A               BUF END (DEFAULT BGN SOURCE)
2421 1194F D6           C=A     A
2422 11951 C9           C=C+B   A               BGN DEST
2423 11953 135          D1=C
2424 11956 440          GOC     IOCN20          (B.E.T.)
2425 11959 E0    IOCN10 A=A-B   A               BGN SOURCE
2426 1195B 6410  IOCN20 GOTO    IOSH20          SHRINK BUFFER
2427                *
2428                *
2429                ********************************************************************
2430                ********************************************************************
2431                **
2432                ** Name:(S) I/OCOL  -  I/O Buffer Collapse
2433                **
```

```
2434            ** Category:   BUFUTL
2435            **
2436            ** Purpose:
2437            **      Collapses specified I/O Buffer -
2438            **      Leaves header intact, but shrinks length to zero
2439            **
2440            ** Entry:
2441            **      C(X) = Buffer ID#
2442            **
2443            ** Exit:
2444            **      Carry clear=> Buffer not found; Created w/zero length
2445            **               set=> Buffer collapsed
2446            **                     D1 past buffer header
2447            **                     P=0
2448            **                     D0 1 nibble past buffer header
2449            **                      (at buffer ID)
2450            **
2451            ** Calls:      I/OFND, MOVEMU, PTRADJ
2452            **
2453            ** Uses.......
2454            **  Inclusive: A-D, D0, D1
2455            **
2456            ** Stk lvls:   2
2457            **
2458            ** Detail:     It is assumed that I/OCOL will only be called
2459            **             on existing buffers; if the buffer doesn't
2460            **             exist, 6 nibbles of user RAM will be utilized
2461            **             for the header w/o the leeway memory check.
2462            **
2463            ** History:
2464            **
2465            **    Date     Programmer             Modification
2466            **  --------   ----------   ----------------------------------
2467            **  07/04/82   S.W.         Added documentation
2468            **
2469            ********************************************************************
2470            ********************************************************************
2471            *
2472            ********************************************************************
2473            ********************************************************************
2474            **
2475            ** Name:(S) I/OALL  -  I/O Buffer Allocate
2476            ** Name:(S) I/OAL+  -  I/O Buffer Allocate
2477            **
2478            ** Category:   BUFUTL
2479            **
2480            ** Purpose:    Allocates space for I/O buffer specified.
2481            **             If it already exists, will expand or contract to
2482            **             conform to size specified.  If it doesn't exist,
2483            **             will create it.
2484            **
2485            ** Entry:      C(X)=ID#
2486            **             B(A)= Desired buffer size (not to exceed FFF)
2487            **
2488            **             I/OALL: Assumes P=0
```

```
2489            **                      Guarantess Leeway added in Mem Check
2490            **              I/OAL+: Sets P=1, guarantess NO Leeway in Mem Ck
2491            **
2492            ** Exit:      CARRY SET   => BUFFER ALLOCATED
2493            **                            D1 points past buffer header
2494            **                            D0 1 nib past buffer header front
2495            **                             (at buffer ID)
2496            **                            P=0
2497            **                            B(A) = buf size if just created,
2498            **                                    else net change in size
2499            **                            C(6-0) contains buf header info:
2500            **                                   C(0)   #addresses to update
2501            **                                   C(1-3) Buf ID
2502            **                                   C(4-6) Buf length
2503            **                            If Buffer already exists and expands
2504            **                               to a larger size:
2505            **                            A=D1 (past buffer header)
2506            **                            D(A) points to point of expansion
2507            **                                 Buffer expanded from bottom
2508            **
2509            **
2510            **                  CLR   => NO ROOM
2511            **                            C(4) = Error Number (eMEM)
2512            **                            P=0
2513            **
2514            ** Calls:     I/OFND, MOVEMD, MEMCL+, MOVEMU, IDLNSV
2515            **
2516            ** Uses:      A, B, C, D, D1, D0, XM, SB
2517            **            C(S) used to save Leeway setting for MEMCL+
2518            **
2519            ** Stack lvls: 3 if existing buffer decreases in size
2520            **             2 otherwise
2521            **
2522            ** History:
2523            **
2524            **    Date      Programmer   Modifications
2525            **  --------    ----------   -------------
2526            **  07/04/82    S.W.         Added documentation
2527            **  09/12/82    J.P.         MEMCL+ interface, entries
2528            **  10/12/82    S.W.         Eliminated I/OAL1 & I/OAL2
2529            **                            entry points. Changed I/OALL
2530            **                            entry point to ASSUME P clear
2531            **  09/13/83    S.W.         Modified stack level doc.
2532            **
2533            ********************************************************************
2534            ********************************************************************
2535            *
2536 1195F 137   IOSH10 CD1EX
2537 11962 06          RSTK=C                BUFST
2538 11964 C2          C=C+A  A              PT TO BUF END (BGN SOURCE MOVEMU)
2539 11966 134         D0=C
2540 11969 DA          A=C    A              BEGIN SOURCE
2541 1196B C9          C=C+B  A              BGN DEST. (PROPOSED BUF END)
2542 1196D 135         D1=C                  BEGIN DEST.
2543 11970 7011  IOSH20 GOSUB  ptradj        SHRINK BUFFER, adjust pointers
```

```
2544 11974 565          GONC   IOEX40          (B.E.T.)
2545               *
2546 11977 D9    IONULL C=B    A               Copy Buffer ID
2547 11979 D1   =I/OCOL B=0    A               Desired buffer length
2548               *
2549 1197B 21   =I/OAL+ P=     1               Set NO Leeway add in MEMCHK
2550               *
2551 1197D 733F =I/OALL GOSUB  I/OFN+          Does CPEX 14 and Sets P=0
2552 11981 DD           CBEX   A
2553 11983 73A0         GOSUB  IDLNSV          SAVE BUF ID & LENGTH IN D
2554               *        C=B    A            BUF LENGTH
2555 11987 546          GONC   I/OCRE          BUFFER NOT FOUND
2556               *
2557 1198A E8           B=B-A  A               AMT TO EXPAND/CONTRACT BUFFER
2558 1198C 42D          GOC    IOSH10
2559               *
2560               * EXPAND BUFFER
2561               *
2562 1198F 137   IOEX24 CD1EX
2563 11992 C2           C=C+A  A               Expand from bottom
2564 11994 134          DO=C                   BGN SOURCE FOR MOVEMD
2565 11997 E2           C=C-A  A
2566 11999 06    IOEX25 RSTK=C                 SAVE BUFST
2567               *
2568               * C(S) holds Leeway setting on entry
2569               * MEMCL+ expects P to tell "type" of MEMCHK
2570               *
2571 1199B 80DF         P=C    15              Set P = Leeway setting
2572 1199F 8F00         GOSBVL =MEMCL+
           000
2573 119A6 4D3          GOC    I/ORT+          Pop level off stk before rtn
2574               *
2575               * D1 @ AVMEMS on return from MEMCL+
2576               * UPDATE IOBFEN ---> AVMEMS
2577               *
2578 119A9 136          CDOEX
2579 119AC DA           A=C    A               Save Begin Source
2580 119AE 79D0         GOSUB  ptrad2          Adjust IOBFEN-AVMEMS
2581 119B2 184          DO=DO- 5               Position at AVMEMS
2582 119B5 146          C=DATO A
2583 119B8 135          D1=C                   End dest
2584 119BB E9           C=C-B  A               END SOURCE
2585 119BD DE           ACEX   A               A=end source; C=bgn source
2586 119BF 8F00         GOSBVL =MOVED2         EXPAND BUFFER
           000
2587               *
2588               * D1=BGN DEST. (BGN SOURCE+OFFSET)
2589 119C6 133          AD1EX
2590 119C9 E0           A=A-B  A               BGN SOURCE (PT OF EXPANSION)
2591 119CB 07    IOEX40 C=RSTK                 C(A)=BUFST
2592 119CD 135          D1=C                   BUFST IN D1
2593 119D0 134          DO=C
2594 119D3 186          DO=DO- 7
2595 119D6 D6           C=A    A               PT OF EXPANSION
2596 119D8 AFF          CDEX   W               D(A) = PT OF EXPANSION
```

```
2597 119DB 15C6          DATO=C 7              C(0)=#addresses;C(3-1)=Buf ID
2598 119DF 160           D0=D0+ 1              Point to Buffer ID
2599 119E2 02            RTNSC                 C(6-4)=Buf length
2600               *
2601 119E4 D5     I/ORT+ B=C    A              Save error#
2602 119E6 07            C=RSTK                Clean up return stack
2603 119E8 D9            C=B    A              Restore error#
2604 119EA 03     I/ORTN RTNCC
2605               *
2606 119EC D2     I/OCRE C=0    A
2607 119EE 307           LCHEX  7              Buf header length
2608 119F1 C1            B=B+C  A
2609               *
2610 119F3 1F67          D1=(5) =IOBFEN        I/O Buffers End
          5F2
2611 119FA 147           C=DAT1 A
2612 119FD 134           D0=C
2613 11A00 183           D0=D0- 4              PT AT 4 zero nibs (Bgn Source)
2614 11A03 135           D1=C
2615 11A06 172           D1=D1+ 3              Pt to where buf start will be
2616 11A09 137           CD1EX
2617 11A0C 5C8           GONC   IOEX25         (B.E.T.)
2618               *
2619               *
2620               ************************************************************
2621               ************************************************************
2622               **
2623               ** Name:(S) I/OEXP  -  I/O Buffer Expand
2624               ** Name:(S) I/OEX2  -  I/O Buffer Expand
2625               **
2626               ** Category:   BUFUTL
2627               **
2628               ** Purpose:
2629               **      Expand I/O buffer from high memory by the amount
2630               **      specified.
2631               **
2632               **      I/OEXP guarantees that the memory check is done
2633               **      including consideration for leeway.
2634               **
2635               **      I/OEX2 does the memory check without regard to leeway.
2636               **
2637               ** Entry:
2638               **      C(X)= Buffer ID#
2639               **      B(A)= Amount to expand buffer (in nibs)
2640               **            Not to exceed 00FFF
2641               **      2 Entry points:
2642               **      1) I/OEXP - P=0
2643               **      2) I/OEX2 - No additional requirements
2644               **
2645               ** Exit:
2646               **      Carry clear=> Buffer not found
2647               **                    OR No room
2648               **                    OR Buffer size requested too big
2649               **                 set=> Buffer expanded
2650               **                       P=0
```

```
2651            **                  D1 points past buffer header
2652            **                  D0 points 1 nibble past buf header
2653            **                   (at buffer ID)
2654            **                  A(A)=D(A)= Point of expansion
2655            **                        (Old buffer end for I/OEXP)
2656            **
2657            ** Calls:      I/OFND, IDLNSV, MEMCL+, MOVEMD
2658            **
2659            ** Uses:       A-D, D1, D0
2660            **             C(S) saves Leeway setting for MEMCL+
2661            **
2662            ** Stk lvls:   2
2663            **
2664            ** History:
2665            **
2666            **    Date      Programmer            Modification
2667            **  --------    ----------   ----------------------------------
2668            **  07/04/82    S.W.         Added documentation
2669            **  09/12/82    J.P.         Added Leeway Check entries
2670            **
2671            ********************************************************************
2672            ********************************************************************
2673            *
2674 11A0F 21   =I/OEX2 P=     1               Guarantee NO leeway check
2675 11A11 7F9E =I/OEXP GOSUB  I/OFN+          Make sure P=0
2676 11A15 500          RTNNC                 BUFFER NOT FOUND
2677 11A18 DD           CBEX   A              C(A)=AMT TO EXP;B(X)=BUF ID
2678 11A1A A32          C=C+A  X              DESIRED BUFFER SIZE
2679 11A1D 4CC          GOC    I/ORTN
2680 11A20 7600         GOSUB  IDLNSV         SAVE BUF ID & LEN IN D
2681 11A24 E8           B=B-A  A              PTR ADJUST OFFSET
2682 11A26 686F         GOTO   IOEX24
2683            *
2684            ********************************************************************
2685            ********************************************************************
2686            **
2687            ** Name:    IDLNSV  -  Save Buffer ID and Buffer Length
2688            **
2689            ** Category:   LOCAL
2690            **
2691            ** Purpose:
2692            **      Saves Buffer ID & Buffer length parameters in D(5-0)
2693            **
2694            ** Entry:
2695            **      C(A)   =  Buffer length
2696            **      C(S)   =  Number of addresses
2697            **      B(X)   =  Buffer ID
2698            **      C(14)  =  Memck parameter (only for I/OEXP,I/OALL)
2699            **
2700            ** Exit:
2701            **      Carry preserved
2702            **      C(X)   =  Buffer ID
2703            **      B(A)   =  Buffer Length
2704            **      D(0)   =  Number of addresses to update (C(S) on entry)
2705            **      D(3-1) =  Buffer ID
```

```
2706                **      D(6-4) =  Buffer Length
2707                **      C(S)   =  C(14) on entry
2708                **
2709                ** Calls:      none
2710                **
2711                ** Uses.......
2712                **  Inclusive: B(A),C(A), C(S), D
2713                **
2714                ** Stk lvls:   0
2715                **
2716                ** History:
2717                **
2718                **    Date     Programmer             Modification
2719                **  --------   ----------   ----------------------------------
2720                **  07/04/82   S.W.         Added documentation
2721                **  03/10/83   S.W.         Using B(S) to save Leeway Setting
2722                **  03/14/83   M.B.         Packed a byte.
2723                **
2724                ********************************************************************
2725                ********************************************************************
2726                *
2727 11A2A AF7    IDLNSV D=C    W               READ IN BUFFER LENGTH
2728 11A2D F3           DSL    A
2729 11A2F F3           DSL    A
2730 11A31 BF3          DSL    W               MAKE ROOM FOR BUF ID#
2731 11A34 DD           CBEX   A               B(A)=BUF LEN; C(X)=BUF ID
2732 11A36 AB7          D=C    X               READ IN BUFFER ID
2733 11A39 ACF          CDEX   S               Read into D(S) the #addresses,
2734                *                                and C(S)= leeway check.
2735 11A3C 813          DSLC                   Shift into low nibble
2736 11A3F 01           RTN
2737                *
2738                *
2739                ********************************************************************
2740                ********************************************************************
2741                **
2742                ** Name:(S) I/ODAL  -  I/O Buffer Deallocate
2743                **
2744                ** Category:   BUFUTL
2745                **
2746                ** Purpose:    Deallocates an I/O Buffer
2747                **
2748                ** Entry:      C(X)=BUF ID#
2749                **
2750                ** Exit:     CARRY SET=> BUFFER DEALLOCATED
2751                **                       P=0
2752                **                CLR=> BUFFER NOT FOUND
2753                **
2754                ** Calls:    I/OFND, MOVEMU, PTRADJ
2755                **
2756                ** Uses:     A, B, C, D1, D0
2757                **
2758                ** Stack lvls: 2
2759                **
2760                ** History:
```

```
2761                **
2762                **    Date     Programmer   Modifications
2763                **  --------   ----------   -------------
2764                **  07/04/82   S.W.         Added documentation.  Modified
2765                **                           code to return with carry set if
2766                **                           buffer deallocated.
2767                **
2768                ****************************************************************
2769                ****************************************************************
2770                *
2771 11A41 7C7E =I/ODAL GOSUB  IOFNDO          SEARCH WITH UNADULTERATED ID
2772 11A45 490        GOC    IODAL2          GO IF FOUND
2773 11A48 7E6E       GOSUB  I/OFND          SEARCH WITH HIBIT IF ID SET
2774 11A4C 500        RTNNC
2775                *
2776 11A4F 1C6  IODAL2 D1=D1- 7              PT TO HEADER START (BGN DEST)
2777 11A52 137        CD1EX
2778 11A55 135        D1=C                   BEGIN DEST.
2779 11A58 D5         B=C    A               "   "
2780 11A5A 134        D0=C
2781 11A5D 166        D0=D0+ 7               POINT PAST HEADER
2782 11A60 136        CD0EX
2783 11A63 CA         A=A+C  A               BUF END (BGN SOURCE)
2784 11A65 E8         B=B-A  A               OFFSET (NEGATIVE)
2785 11A67 7910       GOSUB  ptradj          DELETE BUFFER, adjust pointers
2786                *
2787 11A6B 02         RTNSC
2788                *
2789                * Entry to expand 5 nibbles in Assign Table
2790                *
2791 11A6D 7A20 =EXPCH# GOSUB  STPASG
2792 11A71 7C9F       GOSUB  I/OEXP
2793 11A75 500        RTNNC                  NOT ENOUGH ROOM, CLEAR CARRY
2794 11A78 DB         C=D    A
2795 11A7A 135        D1=C
2796 11A7D D2         C=0    A
2797 11A7F 145        DAT1=C A               ZERO THE NEW ENTRY
2798 11A82 02         RTNSC                  SET CARRY, RETURN
2799                *
2800 11A84 8F00  ptradj GOSBVL =MOVEUA
           000
2801 11A8B 26    ptrad2 P=     ((AVMEMS)-(IOBFEN))/5
2802 11A8D 1B67       D0=(5) =IOBFEN
           5F2
2803 11A94 8D00       GOVLNG =PTRAD2
           000
2804                *
2805 11A9B D2   =STPASG C=0    A
2806 11A9D 305        LC(1)  5
2807 11AA0 D5         B=C    A
2808 11AA2 3240       LC(3)  =bASSGN
           8
2809 11AA7 03         RTNCC
2810                *
2811                *
```

```
2812                    STITLE OPEN file
2813                ********************************************************************
2814                ********************************************************************
2815                **
2816                ** Name:    OPNCH#  -  Open a Channel # in the Assign Table
2817                **
2818                ** Category:   FILUTL
2819                **
2820                ** Purpose:  Open a channel number in the assign table.
2821                **           This routine will not open an entry in the FIB.
2822                **
2823                ** Entry:
2824                **           CHN#SV = CHANNEL #
2825                **
2826                ** Exit:  Carry set => successfully opened
2827                **            D1 @ The channel # in the assign table
2828                **        Carry clear => Error, insufficient memory
2829                **            C(3-0) = Error code
2830                **
2831                ** Calls: I/OFND, I/OALL, FDCH#, CLOSEF, EXPCH#,
2832                **
2833                ** Uses:
2834                **   Inclusive: A,B,C,D,D0,D1, ST0-4, 8
2835                **
2836                ** Stk lvls: +4
2837                **
2838                ********************************************************************
2839                ********************************************************************
2840                *
2841 11AA9 7EFD =OPNCH# GOSUB  BUFASN          FIND ASSIGN TABLE
2842 11AAD 4B1          GOC    OPN120          TABLE FOUND
2843                * ASSIGN TABLE NOT FOUND, CREATE A NEW ONE
2844 11AB0 77EF         GOSUB  STPASG
2845 11AB4 75CE         GOSUB  I/OALL
2846 11AB8 5A0          GONC   NOROOM
2847 11ABB D2           C=0    A
2848 11ABD 145          DAT1=C A                ZERO THE BUFFER
2849 11AC0 48E          GOC    OPNCH#           START IT ALL OVER AGAIN
2850 11AC3 795C  NOROOM GOSUB  CRTFer           LC(4) =eMEM
2851 11AC7 03           RTNCC
2852                * D1 POINTS PASSED BUFFER HEADER
2853                * A(A) = BUFFER LENGTH
2854 11AC9 7695  OPN120 GOSUB  GTSVC#           GET SAVED CHANNEL #  TO B(B)
2855 11ACD 7BD9         GOSUB  FDCH#            SEARCH CHANNEL # IN ASSIGN TABLE
2856 11AD1 571          GONC   OPN150           CHNL # NOT FOUND,CREATE NEW ENTRY
2857 11AD4 968          ?A=0   B                CHANNEL CLOSED ?
2858 11AD7 02           GOYES  OPN200           IF SO, O.K.
2859                * CHANNEL OPENED, CLOSE IT FIRST
2860 11AD9 172          D1=D1+ 3                ZERO FIB # IN ASSIGN TABLE
2861 11ADC D2           C=0    A
2862 11ADE 14D          DAT1=C B
2863 11AE1 72A5         GOSUB  CLOSEF
2864 11AE5 63CF         GOTO   OPNCH#           START IT ALL OVER AGAIN
2865                * ADD A NEW ENTRY TO ASSIGN TABLE
2866 11AE9 708F  OPN150 GOSUB  EXPCH#
```

```
2867 11AED 55D          GONC   NOROOM
2868 11AF0 7F65         GOSUB  GTSVC#          GET SAVED CHANNEL # TO C(B)
2869 11AF4 14D          DAT1=C B               WRITE CHANNEL # TO ASSIGN TABLE
2870              * D1 POINTS TO NEW ENTRY ADDR IN ASSIGN TABLE
2871 11AF7 02    OPN200 RTNSC
2872              *
2873              *
2874 11AF9 8F00  fpoll+ GOSBVL =SNAPSV          Save A,D,D0,D1
           000
2875 11B00 8C00         GOLONG =FPOLL
           00
2876              *
```

```
2877                    STITLE Get New File Number
2878            ****************************************************************
2879            ****************************************************************
2880            **
2881            ** Name:(S) OPENF   -  Open File
2882            ** Name:    OPENF-  -  Open File
2883            ** Name:    OPENF*  -  Open File
2884            ** Name:    OPNF+   -  Open File
2885            **
2886            ** Category:   FILUTL
2887            **
2888            ** Purpose:  Open a new file in the FIB
2889            **
2890            ** Entry:
2891            **   All:   P    =   0
2892            **
2893            **   OPENF:   D0 points at file spec. in the BASIC statement
2894            **
2895            **   OPENF*:  A, D(S), D(A), R0 set up as on exit from FSPECx
2896            **      R2     =  0 if R2/R3 device assignment info not
2897            **                present
2898            **             =  Device assignment info from FSPECx.
2899            **      R3     =  Device assignment info from FSPECx unless
2900            **                R2 = 0.
2901            **
2902            **   OPNF+:   D1 points at start of file header in memory
2903            **
2904            **   OPENF-:  STMTR0 & STMTR1 has the information as the entry
2905            **            condition specified by the WRTFIB routin.
2906            **      R2     =  0 if R2/R3 device assignment info not
2907            **                present
2908            **             =  Device assignment info from FSPECx.
2909            **      R3     =  Device assignment info from FSPECx unless
2910            **                R2 = 0.
2911            **
2912            ** Exit:
2913            **            P  =   0
2914            **            Carry set => Done successfully
2915            **              A(B) = FIB# of file
2916            **              R1 = the new entry address in FIB
2917            **              S10  =  Set if file has I/O buffer
2918            **              STMTD1 = FIB address of file
2919            **              STMTR0, STMTR1 set to exit conditions of WRTFIB
2920            **              The FIB entry filled with proper information
2921            **            Carry clear => Error
2922            **              C(3-0) = Error code
2923            **              File already opened
2924            **              FIB full
2925            **              Insufficient memory
2926            **              Unrecognized file type
2927            **
2928            ** Calls: FSPECx, POLL, FINDF, DATSTR, I/OFND
2929            **
2930            ** Uses:
2931            **  Inclusive: A,B,C,D,D0,D1,R0,R1,STMTR0,STMTR1,S10
```

```
2932            **
2933            ** Stk lvls: 6 at least (FSPECx takes 5, pFINDf requires 6)
2934            **
2935            ** Note: This routine falls into WRTFIB to write the file
2936            **       information in the FIB.
2937            ****************************************************************
2938            ****************************************************************
2939            *
2940            ****************************************************************
2941            ****************************************************************
2942            **
2943            ** Name:(S) pFINDF  -  Find External File
2944            **
2945            ** Category:   POLL
2946            **
2947            ** Type:       POLL
2948            **
2949            ** Purpose:
2950            **      Find a given file in a given mass memory device
2951            **
2952            ** Should poll be "Handled" (return with XM=0)?:Yes
2953            **
2954            ** Meaning of "Handling" Poll (what does code do if handled?):
2955            **      Return file information about the file.
2956            **
2957            ** Entry conditions for handler (registers, ST, RAM, etc.):
2958            **      B[A] = Poll number.
2959            **      HEX mode.
2960            **      P=0.
2961            **      R0 = First 8 chars of file name
2962            **      R1 = Last 2 chars of file name
2963            **      D(X) = Device address
2964            **      D(S) = Device type
2965            **
2966            **
2967            ** Normal exit conditions from handler if handled (ST, RAM,
2968            ** registers, etc.):
2969            **      Carry clear
2970            **      HEX mode.
2971            **      XM=0.
2972            **
2973            **         R0(0,3) = Starting record #
2974            **         R0(4,6) = Device address
2975            **         R0(7,10)= 0000
2976            **         R0(11,14)= File type
2977            **         R0(15)  = 8
2978            **         R1(0)   = Entry # in the record containing directory
2979            **         R1(1,4) = Record # of directory entry
2980            **         R1(5)   = 0
2981            **         R1(6,9) = # of sectors of file length
2982            **
2983            **
2984            ** Normal exit conditions from handler if not handled (ST, RAM,
2985            ** registers, etc.):
2986            **      Carry clear
```

```
2987                **      HEX mode.
2988                **      XM=1.
2989                **
2990                ** Error exit conditions from handler:
2991                **      Carry set.
2992                **      HEX mode.
2993                **      C[0-3] = Error number.
2994                **
2995                ** Available subroutine levels:   6
2996                **
2997                **
2998                ** What registers/RAM may be used if handled?:
2999                **      A,B,C,D(15,5),D1,R0,R1, P
3000                **
3001                ** What registers/RAM may be used if not handled?:
3002                **      A,B,C,D[15-5],D1,R0,R1,P
3003                **
3004                ** What registers/RAM may be used if error exit:
3005                **      A,B,C,D[15-5], P
3006                **
3007                **
3008                ** History:
3009                **
3010                **    Date      Programmer              Modification
3011                ** --------   ----------   ----------------------------------
3012                ** 04/20/83   SC           Document
3013                **
3014                ********************************************************************
3015                ********************************************************************
3016                *
3017 11B06 76DC =OPENF  GOSUB  fspecx
3018 11B0A 560          GONC   OPENF*
3019 11B0D 6337  OPNERR GOTO   bserr
3020                *
3021 11B11 741A =OPENF* GOSUB  SVFSP+          SAVE FILE SPEC. INFORMATION
3022 11B15 522          GONC   OPNF20
3023                *
3024                * FILE IS IN A HP-IL DEVICE
3025                *
3026 11B18 778B         GOSUB  POLLFF          PUT FILE NAME TO R1 AND POLL
3027 11B1C 71           CON(2) =pFINDF         POLL TO FIND THE FILE
3028 11B1E 411          GOC    ERRTN           ERROR OCCUR
3029 11B21 7D91         GOSUB  OPNF+3
3030 11B25 831          ?XM=0
3031 11B28 C2           GOYES  OPENF-          FILE FOUND
3032                *
3033 11B2A 3300  OPNOFD LC(4)  =eFnFND         FILE NOT FOUND
          00
3034 11B30 03    ERRTN  RTNCC
3035                *
3036 11B32 8C00 =findf  GOLONG =FINDF
          00
3037                *
3038 11B38 76FF OPNF20  GOSUB  findf
3039 11B3C 4DE          GOC    OPNOFD
```

```
3040                *
3041 11B3F 7EF0 =OPNF+  GOSUB  OPNF+1
3042 11B43 480          GOC    OPNF25
3043 11B46 7B4A  TYPER  GOSUB  SVFT90          LC(4) =eFTYPE
3044 11B4A 03           RTNCC
3045 11B4C 7241  OPNF25 GOSUB  OPNF+2
3046 11B50 6300         GOTO   OPNF50
3047                * ENTRY POINT CALLED BY ASSIGN #
3048 11B54          =OPENF-
3049                *
3050 11B54 7A4D  OPNF50 GOSUB  BUFFIB          FIND FIB
3051                * D1 POINTS AT START OF FIB
3052                * SEARCH THE FIB FIRST, CHECK THE DATA START ADDR IN ALL THE
3053                * EXIST FIB ENTRY TO DETECT THE DUPLICATE FILE ENTRY.
3054 11B58 133          AD1EX
3055 11B5B D8           B=A    A               SAVE FIB START ADDR IN B
3056 11B5D 133          AD1EX
3057 11B60 1B17         D0=(5) =STMTRO
          8F2
3058 11B67 15E6         C=DAT0 7               C(6-0) = FILE DATA START ADDR
3059 11B6B 26           P=     6
3060 11B6D 14B   FDENT1 A=DAT1 B
3061 11B70 968          ?A=0   B               REACHED END OF FIB BUFFER ?
3062 11B73 72           GOYES  GF#130          IF SO, OK TO OPEN THE FILE
3063 11B75 17C          D1=D1+ (oFBEGb)-(oFIL#b)
3064 11B78 177          D1=D1+ (oDBEGb)-(oFBEGb)
3065 11B7B 15B6         A=DAT1 7
3066 11B7F 912          ?A=C   WP
3067 11B82 E0           GOYES  FILOPN
3068 11B84 17E          D1=D1+ (oRECLb)-(oDBEGb)
3069 11B87 17F          D1=D1+ (oRLENb)-(oRECLb)
3070 11B8A 17A          D1=D1+ (1FIB)-(oRLENb)
3071 11B8D 5FD          GONC   FDENT1          (B.E.T.)
3072                *
3073 11B90 20    FILOPN P=     0
3074 11B92 3300         LC(4)  =eFOPEN
          00
3075 11B98 03           RTNCC
3076                * SEARCH THE FIB AGAIN TO FIND THE FIRST UNUSED FILE #
3077 11B9A 20    GF#130 P=     0
3078 11B9C D4           A=B    A
3079 11B9E 131          D1=A
3080 11BA1 D1           B=0    A
3081 11BA3 D2           C=0    A
3082 11BA5 31F3         LC(2)  =1FIB
3083 11BA9 B65   GF#150 B=B+1  B
3084 11BAC 43E          GOC    FILOPN          BUFFER FULL, SAY "File Open"
3085 11BAF 14B          A=DAT1 B
3086 11BB2 968          ?A=0   B
3087 11BB5 21           GOYES  GF#170          REACHED END OF BUFFER
3088 11BB7 9E0          ?A>B   B               FOUND AN UNUSED FILE # ?
3089 11BBA D0           GOYES  GF#170
3090 11BBC 133          AD1EX                  MOVE T NEXT ENTRY
3091 11BBF CA           A=A+C  A
3092 11BC1 131          D1=A
```

```
3093 11BC4 54E          GONC   GF#150          (B.E.T.)
3094            *D1 POINTS AT THE INSERT POINT
3095 11BC7 133   GF#170 AD1EX                  SAVE INSERT POINT & FILE # IN R1
3096 11BCA BF0          ASL    W
3097 11BCD BF0          ASL    W
3098 11BD0 AE4          A=B    B
3099 11BD3 101          R1=A
3100 11BD6 D5           B=C    A
3101 11BD8 3230         LC(3)  =bFIB
           8
3102 11BDD 703E         GOSUB  I/OEXP          CREATE A NEW ENTRY IN FIB
3103 11BE1 471          GOC    GF#180          Carry set if buffer is expanded
3104            *
3105            * Try to see if memory low or too many file opened
3106            *
3107 11BE4 D2           C=0    A
3108 11BE6 31F3         LC(2)  =lFIB
3109 11BEA 8E00         GOSUBL =chkspc
           00
3110 11BF0 5F9          GONC   FILOPN          If too many files, say "File Open"
3111            *
3112 11BF3 8C00 =nomem  GOLONG =NOMEM          GOTO  MEMERR
           00
3113            *
3114 11BF9 75AC  GF#180 GOSUB  BUFFIB          FIND FIB START ADDR AGAIN
3115 11BFD D8           B=A    A               B= BUFFER LENGTH
3116 11BFF 119          C=R1
3117 11C02 BF6          CSR    W
3118 11C05 BF6          CSR    W               C= NEW ENTRY ADDRESS
3119 11C08 133          AD1EX                  A= BUFFER START ADDR
3120 11C0B C0           A=A+B  A               A= BUFFER END ADDR
3121 11C0D DF           CDEX   A               C= OLD BUFR END ADDR, D=ENTRY ADDR
3122 11C0F 134          D0=C                   D0= END SOURCE ADDR
3123 11C12 EB           C=C-D  A               C= SOURCE SIZE
3124 11C14 131          D1=A                   D1= DEST END ADDR
3125 11C17 8E00         GOSUBL =MOved3         CREATE GAP FOR THE NEW ENTRY
           00
3126 11C1D 136          CD0EX                  C= ENTRY START ADDR
3127 11C20 D5           B=C    A
3128 11C22 135          D1=C
3129 11C25 D2           C=0    A
3130 11C27 31F3         LC(2)  =lFIB
3131 11C2B 8F00         GOSBVL =WIPOUT         CLEAR THE BUFFER
           000
3132 11C32 D9           C=B    A
3133 11C34 135          D1=C
3134 11C37 111          A=R1
3135 11C3A 149          DAT1=A B               WRITE FILE # TO FIB ENTRY
3136 11C3D 60B0         GOTO   WRTFIB
3137            *
3138            * OPNF+1 SHOULD BE CALLED AS SUBROUTINE,
3139            * IF RETURN WITH CARRY CLEAR, MEANS UNRECOGNIZED FILE TYPE.
3140 11C41 1B87  OPNF+1 D0=(5) (=STMTRO)+7     SAVE PORT # IN STMTRO(7-8)
           8F2
3141 11C48 AFB          C=D    W
```

```
3142 11C4B 14C          DAT0=C B
3143 11C4E 167          D0=D0+ 8              SAVE DEVICE TYPE IN STMTR0(15)
3144 11C51 1544         DAT0=C S
3145 11C55 133          AD1EX
3146 11C58 131          D1=A
3147 11C5B 160          D0=D0+ 1
3148 11C5E 140          DAT0=A A              SAVE FILE START IN STMTR1
3149            *
3150            * COMPUTE DATA STRAT OF A FILE
3151            *
3152 11C61 137   DATSTR CD1EX
3153 11C64 D7           D=C    A
3154 11C66 135          D1=C
3155 11C69 17F          D1=D1+ (oFTYPh)
3156 11C6C 8E1E         GOSUBL FTYPFD         SEARCH TYPE TABLE
           3F
3157 11C72 500          RTNNC                 UNRECOGNIZED TYPE
3158 11C75 135          D1=C                  D1 POINTS AT ENTRY IN TYPE TABLE
3159 11C78 172          D1=D1+ 3              D1 POINTS AT SUB-HEADER LENGTH
3160 11C7B AF2          C=0    W
3161 11C7E 14F          C=DAT1 B
3162 11C81 C3           D=D+C  A
3163 11C83 3102         LC(2)  =oFLENh
3164 11C87 CB           C=C+D  A
3165 11C89 26           P=     6
3166 11C8B A0E          C=C-1  P
3167 11C8E 20           P=     0
3168 11C90 02           RTNSC
3169            *
3170 11C92 1B17  OPNF+2 D0=(5) =STMTR0
           8F2
3171 11C99 15C6         DAT0=C 7              SAVE DATA START IN STMTR0
3172 11C9D 16A          D0=D0+ 11
3173 11CA0 1583         DAT0=A 4              SAVE FILE TYPE IN STMTR0
3174 11CA4 164          D0=D0+ 5
3175 11CA7 142          A=DAT0 A              A = FILE START ADDRESS
3176 11CAA D2           C=0    A
3177 11CAC 3152         LC(2)  =oIMPLh
3178 11CB0 CA           A=A+C  A
3179 11CB2 131          D1=A
3180 11CB5 15F7         C=DAT1 8
3181 11CB9 165          D0=D0+ 6
3182 11CBC 15C7         DAT0=C 8              SAVE REC# & REC.LENGTH IN STMTR1
3183 11CC0 02           RTNSC
3184            *
3185 11CC2 1B17  OPNF+3 D0=(5) =STMTR0
           8F2
3186 11CC9 118          C=R0
3187 11CCC 1547         DAT0=C W              SAVE DATA START TO R0
3188 11CD0 16F          D0=D0+ 16
3189 11CD3 119          C=R1
3190 11CD6 15C5         DAT0=C 6              SAVE FILE START TO R1
3191 11CDA 1F93         D1=(5) (=SCRTCH)+56
           9F2
3192 11CE1 15F7         C=DAT1 8
```

```
3193 11CE5 165          D0=D0+ 6              SAVE # OF RECORDS & RECORD LENGTH
3194 11CE8 15C7         DAT0=C 8               TO STMTR1+6(8 NIBS)
3195 11CEC 03           RTNCC
3196            *
3197            ********************************************************************
3198            ********************************************************************
3199            **
3200            ** Name:(S) pDIDST  -  Poll for Device ID Storage
3201            **
3202            ** Category:   POLL
3203            **
3204            ** Type:       FPOLL
3205            **
3206            ** Purpose:
3207            **      Handler for device ID storage (D1 @ destination point)
3208            **
3209            ** Should poll be "Handled" (return with XM=0)?:Yes
3210            **
3211            ** Meaning of "Handling" Poll (what does code do if handled?):
3212            **      Save the device ID in FIB for the file
3213            **
3214            ** Entry conditions for handler (registers, ST, RAM, etc.):
3215            **      B[A] = Poll number.
3216            **      HEX mode.
3217            **      P=0.
3218            **
3219            **      R2 contains C[W] from SETUP
3220            **      (R2[14] is the device code from FILSPx)
3221            **      R3 contains the device ID/volume label
3222            **
3223            ** Normal exit conditions from handler if handled (ST, RAM,
3224            ** registers, etc.):
3225            **      HEX mode.
3226            **      XM=0.
3227            **
3228            ** Normal exit conditions from handler if not handled (ST, RAM,
3229            ** registers, etc.):
3230            **      HEX mode.
3231            **      XM=1.
3232            **
3233            ** Available subroutine levels: 4
3234            **
3235            ** What registers/RAM may be used if handled?:
3236            **      A-D, D0, D1, P R2-R3
3237            **
3238            ** What registers/RAM may be used if not handled?:
3239            **      A-D, D0, D1, P, R2, R3
3240            **
3241            ** History:
3242            **
3243            **    Date     Programmer              Modification
3244            ** --------   ----------   ----------------------------------
3245            ** 04/20/83   SC           Document
3246            **
3247            ********************************************************************
```

```
3248            ******************************************************************
3249            *
3250            ******************************************************************
3251            ******************************************************************
3252            **
3253            ** Name:(S) pDATLN  -  Compute File Len w/Create Code = 8
3254            **
3255            ** Category:   POLL
3256            **
3257            ** Type:       POLL
3258            **
3259            ** Purpose:
3260            **      Compute the file length of an external file whose
3261            **      create code is 8.
3262            **
3263            ** Should poll be "Handled" (return with XM=0)?: Yes
3264            **
3265            ** Meaning of "Handling" Poll (what does code do if handled?):
3266            **      Return the file length of the external file
3267            **
3268            ** Entry conditions for handler (registers, ST, RAM, etc.):
3269            **      B[A] = Poll number.
3270            **      HEX mode.
3271            **      P=0.
3272            **      D[S] = copy code of the file, but already been shifted
3273            **             left once(top bit lost).
3274            **      The directory entery of the file is copied from the mass
3275            **      memory into SCRTCH RAM (64 nibs)
3276            **
3277            ** Normal exit conditions from handler if handled (ST, RAM,
3278            ** registers, etc.):
3279            **      Carry clear (POLL only).
3280            **      HEX mode.
3281            **      XM=0.
3282            **      A(A) = File length of the file in nibbles.
3283            **
3284            ** Normal exit conditions from handler if not handled (ST, RAM,
3285            ** registers, etc.):
3286            **      Carry clear
3287            **      HEX mode.
3288            **      XM=1.
3289            **
3290            ** Error exit conditions from handler (POLL only):
3291            **      Carry set.
3292            **      HEX mode.
3293            **      C[0-3] = Error number.
3294            **
3295            ** Available subroutine levels: 6
3296            **
3297            **
3298            ** What registers/RAM may be used if handled?:
3299            **      A-D, D0, P
3300            **
3301            ** What registers/RAM may be used if not handled?:
3302            **      A-D, D0, D1, P
```

```
3303            **
3304            ** What registers/RAM may be used if error exit (POLL only)?:
3305            **      A-D, D0, D1, P
3306            **
3307            **
3308            ** History:
3309            **
3310            **    Date      Programmer              Modification
3311            **  --------   ----------   ----------------------------------
3312            **  04/20/83   SC           Document
3313            **
3314            ************************************************************************
3315            ************************************************************************
3316            *
3317            ************************************************************************
3318            ************************************************************************
3319            **
3320            ** Name:(S) WRTFIB  -  Write File Information to FIB
3321            **
3322            ** Category:   FILUTL
3323            **
3324            ** Purpose:  Write file information into File Information Buffer
3325            **
3326            ** Entry:
3327            **      D1     @  Entry address of the file in FIB.  FIB# has
3328            **                   already been written to the entry
3329            **      R2     =  0 if R2/R3 device assignment data are not
3330            **                  present (relevant only for exernal files)
3331            **             =  OTHERWISE, device assignment data from FSPECx
3332            **      R3     =  Device assignment data from FSPECx if R2 # 0
3333            **      STMTR0(0-10)= File data start address
3334            **        If file in RAM/ROM:
3335            **            STMTR0(0-4) = Absolute data start address
3336            **            STMTR0(5-6) = 0F
3337            **            STMTR0(7-10)= Don't care
3338            **        If file in port:
3339            **            STMTR0(7-8) = PORT #
3340            **        If file in HP-IL device:
3341            **            STMTR0(0-3) = Record #
3342            **            STMTR0(4-10)= HP-IL address
3343            **      STMTR0(11-14) = File type
3344            **      STMTR0(15) = Device type
3345            **        0 - Mainframe
3346            **        1 - Independent RAM
3347            **        2 - ROM
3348            **        8 - HPIL
3349            **      STMTR1(0-5) = File start address
3350            **        If file in RAM/ROM, this is the absolute
3351            **            address of the file headrer.
3352            **        If file in HP-IL device, this is the record
3353            **            number and byte number of the LIF directory
3354            **            entry address of the file.
3355            **      STMTR1(6-10) = File length in nibbles if the file
3356            **            copy code = 0.
3357            **      STMTR1(6-9) = File length in # of records if the
```

```
3358                **            file copy code = 1.
3359                **      STMTR1(10-13) = Record length in bytes if the file
3360                **            copy code = 1.
3361                **
3362                ** EXIT:
3363                **      Never returns if unrecorgnized file type
3364                **      R1     =  FIB entry address
3365                **      Carry  =  Set if no error
3366                **
3367                ** Calls:
3368                **
3369                ** Uses:
3370                **   Inclusive:  A,B,C,D,D0,D1,R0,R1,R2,R3  S10
3371                **
3372                ** Stk lvls:  +5
3373                ********************************************************************
3374                ********************************************************************
3375                *
3376                *
3377 11CEE 133    =WRTFIB AD1EX                  SAVE ENTRY ADDRESS IN R1
3378 11CF1 101          R1=A
3379 11CF4 133          AD1EX
3380 11CF7 1B17         D0=(5) =STMTR0
           8F2
3381 11CFE 1567         C=DAT0 W
3382 11D02 16A          D0=D0+ 11
3383 11D05 171          D1=D1+ (oFBF#b)-(oFIL#b)
3384 11D08 84A          ST=0   I/Obuf
3385 11D0B A46          C=C+C  S
3386 11D0E 542          GONC   WRTF20
3387                * The file is in a HP-IL device, try to create an I/O buffer
3388                * for the file.
3389                *
3390 11D11 85A          ST=1   I/Obuf
3391 11D14 7F25         GOSUB  GETBUF
3392 11D18 560          GONC   WRTF10
3393 11D1B 67AD         GOTO   NOROOM
3394                *
3395 11D1F 1BC7  WRTF10 D0=(5) (=STMTR0)+11
           8F2
3396 11D26 119          C=R1
3397 11D29 135          D1=C
3398 11D2C 171          D1=D1+ (oFBF#b)-(oFIL#b)
3399 11D2F 1513         DAT1=A X
3400                *
3401 11D33 172   WRTF20 D1=D1+ (oFTYPb)-(oFBF#b)
3402 11D36 D0           A=0    A
3403 11D38 15A3         A=DAT0 4
3404 11D3C 8E71         GOSUBL FTYPF#          SEARCH THE FILE TYPE IN TABLE
           3F
3405 11D42 4D0          GOC    WRTF25          FILE TYPE FOUND IN TABLE
3406 11D45 111          A=R1
3407 11D48 7943         GOSUB  DELFIB
3408 11D4C 69FD  WRTFER GOTO   TYPER
3409                *
```

```
3410 11D50 DA    WRTF25 A=C    A              DO @ TABLE ENTRY
3411 11D52 132          ADOEX                  .
3412 11D55 16F          DO=DO+ 16            DO @ BASE FILE TYPE
3413 11D58 146          C=DATO A             C  = BASE FILE TYPE
3414 11D5B 15D3         DAT1=C 4             WRITE BASE FILE TYPE TO FIB
3415 11D5F 18C          DO=DO- 16-3          DO @ ENTRY DATA OFFSET
3416 11D62 D2           C=0    A
3417 11D64 14E          C=DATO B             C(B) = DATA OFFSET
3418 11D67 D7           D=C    A             D(A) = DATA OFFSET
3419 11D69 AC9          C=B    S             C(S) = TYPE COUNT
3420 11D6C A4E          C=C-1  S
3421 11D6F 173          D1=D1+ (oPROTb)-(oFTYPb)
3422 11D72 1554         DAT1=C S             WRITE PROTECT CODE
3423 11D76 181          DO=DO- 2             DO @ ENTRY COPY CODE
3424 11D79 14E          C=DATO B
3425 11D7C 170          D1=D1+ (oCOPYb)-(oPROTb)
3426 11D7F 1550         DAT1=C P             WRITE COPY CODE
3427 11D83 1574         C=DAT1 S
3428 11D87 AC7          D=C    S             D(S) = COPY CODE
3429 11D8A 171          D1=D1+ (oDEVCb)-(oCOPYb)
3430 11D8D 130          DO=A                 RESTORE DO TO STMTRO
3431 11D90 163          DO=DO+ 4
3432 11D93 14E          C=DATO B
3433 11D96 1550         DAT1=C P             WRITE DEVICE TYPE
3434 11D9A 170          D1=D1+ (oFBEGb)-(oDEVCb)
3435 11D9D 160          DO=DO+ 1
3436 11DA0 15A5         A=DATO 6
3437 11DA4 1595         DAT1=A 6             WRITE FILE START
3438 11DA8 D2           C=0    A
3439 11DAA 305          LC(1)  5
3440 11DAD E3           D=D-C  A             DATA OFFSET - 5
3441 11DAF AFB          C=D    W             C(S) = COPY CODE
3442 11DB2 A4E          C=C-1  S
3443 11DB5 A4E          C=C-1  S
3444 11DB8 590          GONC   WRTF30
3445            * COPY CODE =1: SUBHEADER LENGTH=DATA OFFSET - 13 NIBS
3446 11DBB 328F         LC(3)  0-8           C(A) = FF8
         F
3447 11DC0 CB           C=C+D  A             SUBTRACT 8 FROM C, LEAVE C(XS)=0
3448 11DC2 81E   WRTF30 CSRB                 TURN INTO BYTES
3449 11DC5 175          D1=D1+ (oSHLNb)-(oFBEGb)
3450 11DC8 14D          DAT1=C B             WRTIE SUBHEADER LENGTH
3451 11DCB 87A          ?ST=1  I/Obuf        IS THIS AN EXTERNAL FILE ?
3452 11DCE F1           GOYES  WRTF40        IF SO, GOTO WRTF40
3453            * GET THE PROTECT CODE AND COPY CODE FROM FILE HEADER
3454 11DD0 D2           C=0    A
3455 11DD2 3141         LC(2)  =oFLAGh
3456 11DD6 C2           C=C+A  A
3457 11DD8 134          DO=C
3458 11DDB 14E          C=DATO B
3459 11DDE 136          CDOEX
3460 11DE1 118          C=RO                 GET ADDR SAVED IN RO PREVIOUSLY
3461 11DE4 136          CDOEX
3462 11DE7 163          DO=DO+ (oPROTb)-(oFTYPb)
3463 11DEA 14C          DATO=C B
```

```
3464 11DED 1B17  WRTF40 D0=(5) =STMTRO
           8F2
3465 11DF4 171          D1=D1+ (oDBEGb)-(oSHLNb)
3466 11DF7 15EA         C=DAT0 11
3467 11DFB 15DA         DAT1=C 11              WRITE DATA START
3468 11DFF 86A          ?ST=0  I/Obuf          INTERNAL FILE ?
3469 11E02 91           GOYES  WRTF42          IF SO, GOTO WRTF42
3470 11E04 173          D1=D1+ 4               POLL TO WRITE DEVICE ASSIGN INFO
3471 11E07 7EEC         GOSUB  fpoll+          Save A,D,D0,D1 & FAST POLL
3472 11E0B A0           CON(2) =pDIDST
3473               *
3474 11E0D 7000         GOSUB  =Snaprs         Restore A,D,D0,D1 from snap save
3475               *
3476 11E11 1B17  WRTF41 D0=(5) =STMTRO
           8F2
3477 11E18 1C3          D1=D1- 4
3478 11E1B 17A   WRTF42 D1=D1+ (oREC#b)-(oDBEGb)
3479 11E1E 16F          D0=D0+ 16
3480 11E21 165          D0=D0+ 6
3481 11E24 15E7         C=DAT0 8
3482 11E28 15D7         DAT1=C 8               WRITE # OF RECORDS&RECORD LENGTH
3483 11E2C 17D          D1=D1+ (oDLENb)-(oREC#b)
3484 11E2F 87A          ?ST=1  I/Obuf
3485 11E32 60           GOYES  WRTF45
3486 11E34 67B0         GOTO   WRTF50
3487               *
3488               * File is in a HP-IL device
3489               *
3490 11E38 17A   WRTF45 D1=D1+ (oFSIZb)-(oDLENb)
3491 11E3B 1B22         D0=(5) (=SCRTCH)+33    READ FILE SIZE IN SECTORS FIRST
           9F2
3492 11E42 142          A=DAT0 A
3493 11E45 F4           ASR    A
3494 11E47 164          D0=D0+ 5
3495 11E4A 14A          A=DAT0 B
3496 11E4D 141          DAT1=A A
3497 11E50 1CA          D1=D1- (oFSIZb)-(oDLENb)
3498 11E53 94F          ?D#0   S
3499 11E56 F0           GOYES  DATLN2
3500               * MAINFRAME FILE
3501 11E58 1CD          D1=D1- (oDLENb)-(oREC#b) D1 @ SAVED FILE LENGTH IN NIBS
3502 11E5B 143          A=DAT1 A
3503 11E5E 17D          D1=D1+ (oDLENb)-(oREC#b)
3504 11E61 6280         GOTO   DATL10          (B.E.T.)
3505               *
3506 11E65 A47   DATLN2 D=D+D  S
3507 11E68 541          GONC   DATLN4
3508               * FILE COPY CODE >= 8
3509 11E6B 7C48         GOSUB  poll
3510 11E6F A2           CON(2) =pDATLN
3511 11E71 470          GOC    DTLNER
3512 11E74 831          ?XM=0
3513 11E77 D6           GOYES  DATL10
3514 11E79 62DE  DTLNER GOTO   WRTFER
3515               *
```

```
3516            *
3517 11E7D A47   DATLN4 D=D+D  S
3518 11E80 590          GONC   DATLN6
3519            * FILE COPY CODE = 4 (SIMILIAR TO LIF 1)
3520            *
3521 11E83 F0           ASL    A
3522 11E85 F0           ASL    A
3523 11E87 4A5          GOC    DATLN9          (B.E.T)
3524            *
3525 11E8A A47   DATLN6 D=D+D  S
3526 11E8D 5B3          GONC   DATLN8
3527            * FILE COPY CODE = 2 (HP41 DATA FILE)
3528 11E90 1A93         DO=(4) (=SCRTCH)+56
         9F
3529 11E96 D0           A=0    A
3530 11E98 14A          A=DATO B
3531 11E9B 161          DO=DO+ 2
3532 11E9E F0           ASL    A
3533 11EA0 F0           ASL    A
3534 11EA2 14A          A=DATO B                A(A) = DATA LENGTH IN REGISTERS
3535 11EA5 C4           A=A+A  A
3536 11EA7 C4           A=A+A  A
3537 11EA9 C4           A=A+A  A
3538 11EAB 161          DO=DO+ 2
3539 11EAE 14E          C=DATO B                SEE IF THIS FILE SECURED
3540 11EB1 90A          ?C=0   P
3541 11EB4 E2           GOYES  DATLN9
3542 11EB6 119          C=R1
3543 11EB9 134          DO=C
3544 11EBC 168          DO=DO+ (oPROTb)
3545 11EBF 301          LC(1)  1
3546 11EC2 1540         DATO=C P
3547 11EC6 5B1          GONC   DATLN9
3548            * FILE COPY CODE = 1 (FIX LENGTH DATA FILE)
3549 11EC9 D2    DATLN8 C=0    A
3550 11ECB D0           A=0    A
3551 11ECD 1CD          D1=D1- (oDLENb)-(oREC#b)
3552 11ED0 15F3         C=DAT1 4                C= # OF RECORDS
3553 11ED4 173          D1=D1+ 4
3554 11ED7 15B3         A=DAT1 4                A= RECORD LENGTH
3555 11EDB 179          D1=D1+ (oDLENb)-(oRECLb)
3556 11EDE 7409         GOSUB  a-mult
3557            *
3558 11EE2 C4    DATLN9 A=A+A  A                CONVERT BYTES TO NIBS
3559 11EE4 D6    DATL10 C=A    A
3560 11EE6 D3           D=0    A                DON'T SUBTRACT SUB.HRD.LEN
3561 11EE8 6710         GOTO   WRTF60
3562            *
3563            * FILE IS IN RAM/ROM
3564            *
3565 11EEC D2    WRTF50 C=0    A
3566 11EEE 3102         LC(2)  =oFLENh
3567 11EF2 CA           A=A+C  A
3568 11EF4 130          DO=A                    DO  @ FILE CHAIN LENGTH
3569 11EF7 142          A=DATO A                A =FILE CHAIN LENGTH
```

```
3570 11EFA 3150          LC(2)  5
3571 11EFE EE            C=A-C  A
3572 11F00 EB     WRTF60 C=C-D  A              SUBTRACT SUBHDR LEN. FROM FILE LEN
3573 11F02 145           DAT1=C A              C(A) = FILE DATA LENGTH IN NIBS
3574 11F05 1C9           D1=D1- (oDLENb)-(oRECLb)
3575 11F08 147           C=DAT1 A              READ RECORD LENGTH
3576 11F0B 17F           D1=D1+ (oRLENb)-(oRECLb)
3577 11F0E 145           DAT1=C A
3578 11F11 1CB           D1=D1- (oRLENb)-(oCPOSb)
3579 11F14 AF2           C=0    W              SET CURRENT POSITION TO FFFFF
3580 11F17 15D5          DAT1=C 6
3581 11F1B CE            C=C-1  A
3582 11F1D 145           DAT1=C A
3583 11F20 111           A=R1
3584 11F23 8C04          GOLONG REWIND         REWIND THE FILE TO START OF DATA
          4F
3585               *
3586               ***************************************************************
3587               ***************************************************************
3588               ** Name:    ASSIGN  -  ASSIGN# statement execution
3589               **
3590               ** Category:  STEXEC
3591               **
3592               ** Purpose:  Assign a channel number to a file.
3593               **
3594               ** Entry:
3595               **       DO past the ASSIGN# token
3596               **
3597               ** Exit:
3598               **       Exit to next statement execution (NXTSTM)
3599               **
3600               ** Calls:  GETCH#, POLL, OPEN#, CLOSE#, GFIL#, WRTFIB
3601               **
3602               ***************************************************************
3603               ***************************************************************
3604               *
3605 11F29 0000          REL(5) =ASSNDC
          0
3606 11F2E 0000          REL(5) =ASSNP
          0
3607 11F33 8EEE =ASSIGN GOSUBL GETCH#         GET CHHANNEL # AND SAVE IT
          4F
3608 11F39 161           DO=DO+ 2              PASS THE tTO TOKEN
3609 11F3C 14A           A=DATO B
3610 11F3F 3100          LC(2)  =t*
3611 11F43 966           ?A#C   B
3612 11F46 01            GOYES  ASN050
3613               * ASSIGN # TO *:  CLOSE THE CHANNEL
3614 11F48 7711  ASN020 GOSUB  GTSVC#         GET THE SAVED CHANNEL # TO B(B)
3615 11F4C 7121          GOSUB  CLOSE#
3616 11F50 8CC0  ASNRTN GOLONG CTE300
          7F
3617               *
3618 11F56 7688  ASN050 GOSUB  fspecx
3619 11F5A 504           GONC   ASN060
```

```
3620                *
3621                * CHECK IF THE FILE SPEC. IS "*" OR "" IF ERROR RETURN
3622                *
3623 11F5D 867              ?ST=0  7              STRING EXPR ?
3624 11F60 B7              GOYES  bserr1         IF NOT, ERROR
3625 11F62 8E00            GOSUBL =csrc5         SAVE THE ERROR NUMBER
           00
3626 11F68 8E00            GOSUBL =D1=AVE        D1 @ TOP OF STACK
           00
3627 11F6E 171             D1=D1+ 2
3628 11F71 143             A=DAT1 A
3629 11F74 D8              B=A    A
3630 11F76 CD              B=B-1  A              NULL STRING ?
3631 11F78 4FC             GOC    ASN020         IF SO, GORO ASN020
3632 11F7B 17D             D1=D1+ 14
3633 11F7E 31A2            LCASC  \*\
3634 11F82 143             A=DAT1 A
3635 11F85 966             ?A#C   B
3636 11F88 90              GOYES  ASN055
3637 11F8A CD              B=B-1  A
3638 11F8C CD              B=B-1  A
3639 11F8E 49B             GOC    ASN020
3640 11F91 8E00  ASN055    GOSUBL =Cslc5
           00
3641 11F97 6340            GOTO   bserr1
3642                *
3643 11F9B 8E88  ASN060    GOSUBL SVFSP+         SAVE FILE SPEC
           5F
3644 11FA1 5D1             GONC   ASN100
3645                * POLL HP-IL ROM TO ASSIGN A CHANNEL # TO A FILE
3646                *   PROCESS # = pFINDF
3647                *   EXIT:
3648                *
3649 11FA4 8E9F            GOSUBL POLLFF         PUT FILE NAME TO R1 AND POLL
           6F
3650 11FAA 71              CON(2) =pFINDF
3651 11FAC 4E2             GOC    bserr1
3652                *
3653 11FAF 831   ASN070    ?XM=0
3654 11FB2 50              GOYES  ASN80
3655 11FB4 522             GONC   ASNerr
3656 11FB7 770D  ASN80     GOSUB  OPNF+3         WRITE DATA START & FILE START
3657 11FBB 6870            GOTO   ASN200           TO R0 & R1
3658                * INTERNAL FILE
3659 11FBF 7F6B  ASN100    GOSUB  findf
3660 11FC3 5D5             GONC   ASN120         FILE FOUND
3661                *
3662                * FILE NOT FOUND, LET'S CREATE IT AS A DATA FILE
3663                *
3664 11FC6 1B58            D0=(5) (=STMTR1)+4
           8F2
3665 11FCD 1564            C=DAT0 S
3666 11FD1 B46             C=C+1  S              TURN F INTO 0
3667 11FD4 4A0             GOC    ASN110         ASSIGN TO MAINFRAME FILE ?
3668 11FD7 7F4B  ASNerr    GOSUB  OPNOFD
```

```
3669 11FDB 6562  bserr1 GOTO   bserr
3670           *
3671 11FDF 165   ASN110 D0=D0+ 6
3672 11FE2 350F         LCHEX  11E0F0          COPY CODE, CREATE CODE & FILE TYPE
           0E11
3673 11FEA 15C5         DAT0=C 6               SAVE IT IN S-R1-2(10-15)
3674 11FEE AF2          C=0    W
3675 11FF1 31D2         LC(2)  (=oIMPLh)+8
3676 11FF5 AF3          D=0    W
3677 11FF8 8F00         GOSBVL =CREATF         TRY TO CREATE THE FILE
           000
3678 11FFF 4BD          GOC    bserr1          Carry set if unsuccessfully
3679           *
3680 12002 8EC5         GOSUBL CRFSB-          WRITE THE FILE HEADER
           6F
3681 12008 D2           C=0    A
3682 1200A 15D3         DAT1=C 4
3683 1200E 173          D1=D1+ 4
3684 12011 B26          C=C+1  XS
3685 12014 15D3         DAT1=C 4
3686 12018 AF3          D=0    W
3687 1201B 111          A=R1
3688 1201E 131          D1=A                   D1= FILE HEADER ADDRESS
3689 12021 7C1C  ASN120 GOSUB  OPNF+1
3690 12025 4A0          GOC    ASN150
3691 12028 7A1B         GOSUB  TYPER
3692 1202C 6412  ASN130 GOTO   bserr
3693 12030 7E5C  ASN150 GOSUB  OPNF+2
3694           *
3695 12034 7B20  ASN200 GOSUB  GTSVC#          GET SAVED CHANNEL # TO B(B)
3696 12038 7D6A         GOSUB  OPNCH#          OPEN THE CHANNEL #
3697 1203C 5FE          GONC   ASN130
3698 1203F 711B         GOSUB  OPENF-
3699 12043 58E          GONC   ASN130
3700           *
3701 12046 7910  ASN300 GOSUB  GTSVC#          GET SAVED CHANNEL # TO B(B)
3702 1204A 8EC5         GOSUBL FDCH#           FIND THE CHANNEL # AGAIN
           4F
3703 12050 172          D1=D1+ 3
3704 12053 111          A=R1
3705 12056 130          D0=A                   D0 POINTS AT FIB ENTRY
3706 12059 14E          C=DAT0 B               C= FIB #
3707 1205C 14D          DAT1=C B
3708 1205F 60FE         GOTO   ASNRTN
3709           *
3710           *
3711 12063 1BF6 =GTSVC# D0=(5) =CHN#SV
           9F2
3712 1206A 14E          C=DAT0 B
3713 1206D D5           B=C    A
3714 1206F 01           RTN
3715           *
3716           *
```

```
3717                    STITLE CLOSE file
3718                ********************************************************************
3719                ********************************************************************
3720                **
3721                ** Name:    CLOSE#  -  Close File
3722                ** Name:(S) CLOSEF  -  Close File
3723                **
3724                ** Category:   FILUTL
3725                **
3726                ** Purpose:  Close file in File Information Buffer
3727                **
3728                ** Entry:
3729                **     CLOSE#:  B(B) = Channel # of the file
3730                **     CLOSEF:  A(B) = FIB # of the file
3731                **
3732                ** Exit:
3733                **       No error condition if the file not found
3734                **
3735                ** Calls:    FFIB#, POLL
3736                **
3737                ** Uses:     A,B,C,D0,D1, STMTD1
3738                **
3739                ** Stk lvls: 5
3740                **
3741                ** NOTE:  This program FALLS INTO routine DELFIB
3742                **
3743                ********************************************************************
3744                ********************************************************************
3745                *
3746 12071 8E53 =CLOSE# GOSUBL FDCH#          SEARCH CHANNEL # IN ASSIGN TABLE
           4F
3747 12077 500          RTNNC                IF NOT FOUND, DONE
3748 1207A 968          ?A=0   B             CHANNEL CLOSED ?
3749 1207D 00           RTNYES               IF SO, DONE
3750 1207F D2           C=0    A
3751 12081 172          D1=D1+ 3
3752 12084 14D          DAT1=C B             CLOSE THE CHANNEL
3753                *
3754 12087 7462 =CLOSEF GOSUB  FFIB#         Find File# IN FIB
3755 1208B 400          RTNC                 File# NOT FOUND, DONE
3756                * D1 POINTS AT THE FILE ENTRY IN FIB
3757 1208E 133          AD1EX                SAVE FIB ENTRY ADDR IN A
3758 12091 7271         GOSUB  FLSHIO        FLUSH FILE I/O BUFFER IF NEED TO
3759                *
3760                ********************************************************************
3761                ********************************************************************
3762                **
3763                ** Name:    DELFIB  -  Delete FIB Entry
3764                **
3765                ** Category:   FILUTL
3766                **
3767                ** Purpose:
3768                **      Delete an FIB entry for a file
3769                **
3770                ** Entry:
```

```
3771            **      A(A)   =  Address of FIB entry to be deleted
3772            **
3773            ** Exit:
3774            **      Carry clear
3775            **
3776            ** Calls:      IOCNDO
3777            **
3778            ** Uses.......
3779            **  Exclusive: C(A),B(A),D0,R0
3780            **  Inclusive: As above, plus IOCNDO
3781            **
3782            ** Stk lvls:   +4
3783            **
3784            ** NOTE:
3785            **      Routine CLOSEF FALLS INTO this routine!
3786            **
3787            ** History:
3788            **
3789            **    Date      Programmer            Modification
3790            **  --------    ----------   ---------------------------------
3791            **  09/12/82    S.C.         Designed and coded
3792            **  09/13/83    S.W.         Changed doc. to reflect stk lvls=4
3793            **
3794            ********************************************************************
3795            ********************************************************************
3796 12095 130  =DELFIB D0=A                  Set D0 to start of contraction
3797 12098 100          R0=A                  SAVE PTR IN R0 TOO
3798 1209B 161          D0=D0+ (oFBF#b)
3799 1209E 146          C=DAT0 A
3800 120A1 93A          ?C=0   X              IS THERE AN I/O BUFFER ?
3801 120A4 72           GOYES  DELBF8         IF NOT, JUST DELETE THE FIB ENTRY
3802 120A6 7799         GOSUB  I/ODAL         DEALLOCATE THE FILE I/O BUFR
3803            * IF THE ASSIGN TYPE OF THIS DEVICE IS 4, DELETE THE I/O BUFFER
3804            * USED FOR STORING THE DEVICE ID
3805 120AA 110   DELBF5 A=R0
3806 120AD 130          D0=A
3807 120B0 16C          D0=D0+ (oFBEGb)
3808 120B3 16F          D0=D0+ (oDBEGb)-(oFBEGb)+8
3809 120B6 14A          A=DAT0 B
3810 120B9 304          LC(1)  4
3811 120BC 906          ?A#C   P
3812 120BF C0           GOYES  DELBF8
3813 120C1 160          D0=D0+ 1
3814 120C4 146          C=DAT0 A              C(X) = I/O BUFFER ID
3815 120C7 7679         GOSUB  I/ODAL
3816 120CB 110   DELBF8 A=R0
3817 120CE 130          D0=A
3818 120D1 D2           C=0    A
3819 120D3 31F3         LC(2)  =lFIB
3820 120D7 D5           B=C    A
3821 120D9 3230         LC(3)  =bFIB
             8
3822 120DE 7348         GOSUB  IOCNDO
3823 120E2 03           RTNCC
3824            *
```

```
3825            ********************************************************************
3826            ********************************************************************
3827            **
3828            ** Name:(S) CLOSEA  -  Close All Open Files
3829            **
3830            ** Category:   FILUTL
3831            **
3832            ** Purpose:  Close all opening files and delete their entries
3833            **
3834            ** Entry:  P = 0
3835            **
3836            ** Exit:  P = 0
3837            **
3838            ** Calls:  I/OFND, DELFIB, POLL(pWRCBF)
3839            **
3840            ** Uses:  A-D, D0, D1, R0 ,STMTD1
3841            **
3842            ** Stk lvls:  5
3843            **
3844            ********************************************************************
3845            ********************************************************************
3846            *
3847 120E4 8E8B =CLOSEA GOSUBL BUFFIB
           7F
3848 120EA 581          GONC   DELASN
3849 120ED 14F          C=DAT1 B
3850 120F0 96A          ?C=0   B              REACHED END OF FIB YET ?
3851 120F3 00           RTNYES
3852 120F5 133          AD1EX
3853 120F8 7B01         GOSUB  FLSHIO
3854 120FC 759F         GOSUB  DELFIB
3855 12100 53E          GONC   CLOSEA
3856            * DELETE THE ENTIRE ASSIGNMENT TABLE
3857 12103 8E2A  DELASN GOSUBL BUFASN
           7F
3858 12109 500          RTNNC
3859 1210C 6439  IODALj GOTO   I/ODAL
3860            *
3861            ********************************************************************
3862            ********************************************************************
3863            **
3864            ** Name:    DLFIBA  -  Delete All FIB Entries
3865            **
3866            ** Category:   FILUTL
3867            **
3868            ** Purpose:  Delete all FIB entries but will not flush the I/O
3869            **           buffer.
3870            **
3871            ** Entry:  P = 0
3872            **
3873            ** Exit:  P = 0
3874            **
3875            ** Uses:  A-D, D0, D1, R0
3876            **
3877            ** Stk lvls:  5
```

```
3878                **
3879                ************************************************************************
3880                ************************************************************************
3881                *
3882 12110 8EC8 =DLFIBA GOSUBL BUFFIB
           7F
3883 12116 5CE          GONC   DELASN
3884 12119 14F          C=DAT1 B
3885 1211C 96A          ?C=0   B              REACHED END OF FIB YET ?
3886 1211F 4E           GOYES  DELASN
3887 12121 133          AD1EX
3888 12124 7D6F         GOSUB  DELFIB
3889 12128 57E          GONC   DLFIBA         (B.E.T.)
3890                *
3891                ************************************************************************
3892                ************************************************************************
3893                **
3894                ** Name:    FIBON   -  Reset Devices, Buffers at Power On/Off
3895                ** Name:(S) FIBOFF  -  Reset Devices, Buffers at Power On/Off
3896                **
3897                ** Category:   FILUTL
3898                **
3899                ** Purpose: When HP-71 powers off, reset all external devices.
3900                **          When HP-71 powers on, reclaim all the I/O buffers.
3901                **
3902                ** Entry: None
3903                **
3904                ** Exit: P=0.
3905                **       Hex mode.
3906                **
3907                ** Calls: I/OFND, I/ORES
3908                **
3909                ** Uses: A,C, D0,D1, S0
3910                **
3911                ** Stk lvls: 2
3912                ************************************************************************
3913                ************************************************************************
3914                *
3915 1212B 850   =FIBON  ST=1   0
3916 1212E 6600         GOTO   FIBO/F
3917 12132 840  =FIBOFF ST=0   0
3918                *
3919 12135 8E76  FIBO/F GOSUBL BUFFIB
           7F
3920 1213B 133          AD1EX
3921 1213E 130          D0=A
3922 12141 14E   RCLM10 C=DAT0 B
3923 12144 96A          ?C=0   B              REACHED END OF FIB ?
3924 12147 00           RTNYES
3925 12149 161          D0=D0+ (oFBF#b)
3926 1214C 146          C=DAT0 A
3927 1214F 16A          D0=D0+ (oFBEGb)-(oFBF#b)
3928 12152 16B          D0=D0+ (oDBEGb)-(oFBEGb)+4
3929 12155 93A          ?C=0   X              EXTERNAL FILE ?
3930 12158 43           GOYES  NXTFIL         IF NOT, NEXT FILE
```

```
3931 1215A 870          ?ST=1  0              POWER ON ?
3932 1215D D0           GOYES  RCLM20         IF SO, RECLAIM THE I/O BUFFERS
3933 1215F D2           C=0    A
3934 12161 CE           C=C-1  A
3935 12163 1543         DATO=C X
3936 12167 442          GOC    NXTFIL         (B.E.T.)
3937 1216A 8EF8  RCLM20 GOSUBL I/ORES
          7F
3938 12170 162          D0=D0+ 3
3939 12173 14A          A=DATO B
3940 12176 304          LC(1)  4
3941 12179 906          ?A#C   P
3942 1217C D0           GOYES  RCLM30
3943 1217E 146          C=DATO A
3944 12181 F6           CSR    A
3945 12183 8E67         GOSUBL I/ORES
          7F
3946 12189 182   RCLM30 D0=D0- 3
3947               *
3948 1218C 16E   NXTFIL D0=D0+ (oCPOSb)-(oDBEGb)-4
3949 1218F 16B          D0=D0+ (oRLENb)-(oCPOSb)
3950 12192 16A          D0=D0+ (1FIB)-(oRLENb)
3951 12195 5BA          GONC   RCLM10
3952               *
3953               ********************************************************************
3954               ********************************************************************
3955               **
3956               ** Name:(S) PUGFIB  -  Purge the FIB Entries of Purged Files
3957               **
3958               ** Category:   FILUTL
3959               **
3960               ** Purpose: Purge the FIB entry of a purged file.
3961               **          Delete an FIB entry whose "File Begin" address
3962               **          is zero.
3963               **
3964               ** Entry:   The "file Begin" of the purged file in FIB
3965               **           should be already zeroed.
3966               **
3967               ** Exit: The first FIB entries matching the condition is deleted
3968               **
3969               ** Calls: FDFIL#, DELFIB, I/ODAL
3970               **
3971               ** Uses: A-D,D0,D1,R0
3972               **
3973               ** Stk lvls: +4
3974               ********************************************************************
3975               ********************************************************************
3976               *
3977 12198 8E40 =PUGFIB GOSUBL BUFFIB         GET FIB BUFFER ADDRESS
          7F
3978               *
3979 1219E 14B   PUFB10 A=DAT1 B
3980 121A1 968          ?A=0   B
3981 121A4 00           RTNYES
3982 121A6 17C          D1=D1+ (oFBEGb)
```

```
3983 121A9 147          C=DAT1 A
3984 121AC 1CC          D1=D1- (oFBEGb)
3985 121AF 8AA          ?C=0   A
3986 121B2 11           GOYES  CLSCH#
3987 121B4 17F          D1=D1+ 16
3988 121B7 17F          D1=D1+ 16
3989 121BA 17F          D1=D1+ 16
3990 121BD 17E          D1=D1+ (1FIB)-48      PTS TO NEXT ENTRY
3991 121C0 5DD          GONC   PUFB10
3992              *
3993 121C3 D8    CLSCH# B=A    A              B(B) = FIB # OF THIS FILE
3994 121C5 137          CD1EX
3995 121C8 D7           D=C    A              D(A) = FIB ENTRY ADDRESS
3996 121CA 8E82         GOSUBL FDFIL#         FIND THE FILE # IN ASSIGN TABLE
         3F
3997 121D0 570          GONC   PUFB20         IF NOT FOUND, DON'T CLEAR CHANNEL #
3998 121D3 D0           A=0    A              ZERO THE FILE # OF THE CHANNEL #
3999 121D5 149          DAT1=A B
4000 121D8 DF    PUFB20 CDEX   A
4001 121DA DA           A=C    A              A(A) = FIB ENTRY ADDRESS
4002 121DC 68BE         GOTO   DELFIB         NOW DELETE THE FIB ENTRY
4003              *
4004              ********************************************************************
4005              ********************************************************************
4006              ** Name:    FLUSHA  -  Flush All File I/O Buffers
4007              **
4008              ** Category:   FILUTL
4009              **
4010              ** Purpose:  Writes the content of all the file I/O buffers back
4011              **           to the file if buffer content has been altered.
4012              **
4013              ** Entry:  P = 0
4014              **
4015              ** Exit:   P = 0
4016              **
4017              ** Calls:  I/OFND, DELFIB, IOCNDO, POLL(pWRCBF)
4018              **
4019              ** Uses:   A-D, D0, STMTD1
4020              **
4021              ** Stk lvls:  3
4022              **
4023              ********************************************************************
4024              ********************************************************************
4025              *
4026 121E0 8ECB =FLUSHA GOSUBL BUFFIB
         6F
4027 121E6 500          RTNNC
4028 121E9 133          AD1EX                 A= FILE ENTRY ADDRESS IN FIB
4029 121EC 131   FLHA10 D1=A
4030 121EF 14F          C=DAT1 B
4031 121F2 96A          ?C=0   B              REACHED END OF FIB BUFFER ?
4032 121F5 00           RTNYES                IF SO, DONE
4033 121F7 7C00         GOSUB  FLSHIO         FLUSH FILE I/O BUFFER IF NEED TO
4034 121FB D2    FLHA20 C=0    A
4035 121FD 31F3         LC(2)  =1FIB          JUMP OVER INTERNAL FILE
```

```
4036 12201 CA          A=A+C  A
4037 12203 68EF        GOTO   FLHA10          LOOK AT NEXT FILE
4038           *
4039           *
4040           ************************************************************************
4041           ************************************************************************
4042           **
4043           ** Name:    FLSHIO  -  Flush Out File I/O Buffer
4044           **
4045           ** Category:   FILUTL
4046           **
4047           ** Purpose:  Flush file I/O buffer for a given file
4048           **
4049           ** Entry:  A(A):   Contains the FIB entry address of the file
4050           **
4051           ** Exit:  Carry set => The file is internal or the contents of
4052           **                     the file I/O buffer have not been altered.
4053           **        Carry clear => The file I/O buffer has been flushed.
4054           **        If HP-IL error happen, exit to error routine.
4055           **
4056           ** Calls:  POLL(pWRCBF)
4057           **
4058           ** Uses:  B,C,D0, STMTD1
4059           **
4060           ** Stk lvls:  4
4061           **
4062           ************************************************************************
4063           ************************************************************************
4064           **
4065 12207 1B69 =FLSHIO D0=(5) =STMTD1       SAVE FIB ENTRY ADDR IN STMTD1
          8F2
4066 1220E 140         DAT0=A A
4067 12211 130         D0=A
4068 12214 161         D0=D0+ (oFBF#b)
4069 12217 146         C=DAT0 A
4070 1221A 93A         ?C=0   X              IS THIS AN EXTERNAL FILE ?
4071 1221D 00          RTNYES                IF NOT, RETURN CARRY SET
4072 1221F 168         D0=D0+ (oACCSb)-(oFBF#b)
4073 12222 14E         C=DAT0 B
4074 12225 90A         ?C=0   P              ACCESS CODE = 0 ?
4075 12228 00          RTNYES                IF SO, RETURN CARRY SET
4076 1222A 07          C=RSTK                Save on elevel of RSTK
4077 1222C 79C8        GOSUB  fpoll+         Save A,D,D0,D1 & Fast Poll
4078 12230 A1          CON(2) =pWRCBF
4079 12232 D9          C=B    A
4080 12234 06          RSTK=C                Restore last RSTK
4081 12236 831  =CHKRSP ?XM=0                HP-IL MODULE PLUG-IN ?
4082 12239 A2          GOYES  GTBF25         IF SO, RETURN CARRY CLEAR
4083 1223B 3300        LC(4)  =eDVCNF        SAY DEVICE NOT FOUND
          00
4084 12241 8C00  bserr GOLONG =BsErr         ERROR EXIT
          00
4085           *
4086           *
4087           ************************************************************************
```

```
4088            ****************************************************************
4089            **
4090            ** Name:    GETBUF  -  Get I/O Buffer for External File
4091            **
4092            ** Category:   FILUTL
4093            **
4094            ** Purpose:  Get an I/O buffer for an external file
4095            **
4096            ** Entry:  P = 0
4097            **
4098            ** Exit:   Carry clear => found an unused buffer
4099            **          A(x) = I/O buffer number
4100            **          The I/O buffer is created
4101            **         Carry set => No available I/O buffer found
4102            **
4103            ** Calls:  IOFSCR, I/OALL
4104            **
4105            ** Uses:  A, B, C, D, D0, D1
4106            **
4107            ** Stk lvls:  3
4108            **
4109            ** Notes:  Available I/O buffers are from #E00 - #FFF
4110            **         total 512 buffers.
4111            **
4112            ** History:
4113            **
4114            **    Date     Programmer   Modifications
4115            **  --------   ----------   -------------
4116            **  02/09/83   S.W.         Cut code by calling IOFSCR
4117            **
4118            ****************************************************************
4119            ****************************************************************
4120            **
4121 12247 8E14 =GETBUF GOSUBL IOFSCR         Look for available buffer
           6F
4122 1224D 400          RTNC                 None available (not likely)
4123 12250 D1           B=0    A
4124 12252 B25          B=B+1  XS
4125 12255 C5           B=B+B  A             SIZE OF 512 NIBBLES
4126 12257 8E02         GOSUBL I/OALL        ALLOCATE THE BUFFER
           7F
4127 1225D 570          GONC   GTBF30        NOT ENOUGH TO ALLOCATE THE BUFFER
4128 12260 142          A=DATO A             A(X) = BUFFER ID
4129 12263 03    GTBF25 RTNCC
4130 12265 02    GTBF30 RTNSC
4131            *
4132            ****************************************************************
4133            ****************************************************************
4134            **
4135            ** Name:    POPCH#  -  Pop Lowest Assign Table Channel Number
4136            **
4137            ** Category:   FILUTL
4138            **
4139            ** Purpose:  Pop off the lowest level of channel # from the
4140            **           Assign Table
```

```
4141                **
4142                ** Entry:  None
4143                **
4144                ** Exit:   P=0.
4145                **
4146                ** Calls: FNDMRK, I/OFND, CLOSEF, I/OCON, I/ODAL
4147                **
4148                ** Used:  Inclusive A-D, D0,D1, S3
4149                **
4150                ** Stk lvls: 4
4151                ****************************************************************
4152                ****************************************************************
4153                *
4154 12267 7660 =POPCH# GOSUB  FNDMRK
4155 1226B 500         RTNNC                 ASSIG TABLE NOT FOUND
4156 1226E 171         D1=D1+ 2
4157 12271 147         C=DAT1 A              C(X) = LEVEL COUNT
4158 12274 93A         ?C=0   X              DUPLICATE LEVEL EXIST ?
4159 12277 A0          GOYES  POP#10         IF NOT, POP THE ENTIRE LEVEL OFF
4160                * LAST SUB-PROGRAM CALL PAST NO PARAMETER, SO ALL WE HAVE TO DO
4161                * IS TO DECREMENT THE LEVEL COUNT
4162 12279 CE          C=C-1  A
4163 1227B 1553        DAT1=C X
4164 1227F 03          RTNCC
4165                * POP OFF ONE CHANNEL # AT A TIME
4166 12281 8E42  POP#10 GOSUBL BUFASN
           6F
4167 12287 137         CD1EX
4168 1228A C2          C=A+C  A
4169 1228C 135         D1=C                  D1 PTS PAST END OF BUFFER
4170 1228F 1C4         D1=D1- 5
4171 12292 147         C=DAT1 A              READ LAST ENTRY
4172 12295 843         ST=0   3
4173 12298 96A         ?C=0   B              REACHED LEVEL MARK YET ?
4174 1229B 72          GOYES  POP#30         IF SO, GOTO POP#30
4175 1229D 853         ST=1   3
4176 122A0 92E         ?C#0   XS             IS THIS AN INDIRECT CHANNEL # ?
4177 122A3 C0          GOYES  POP#20         IF SO, JUST POP OFF THE ENTRY
4178 122A5 172         D1=D1+ 3
4179 122A8 14B         A=DAT1 B              READ THE FIB #
4180 122AB 78DD        GOSUB  CLOSEF         CLOSE THE FILE
4181                * CONTRACT THE BUFFER BY 5 NIBBLES
4182 122AF 8E6E  POP#20 GOSUBL STPASG
           7F
4183                *
4184 122B5 8E56  POP#25 GOSUBL I/OCON
           6F
4185 122BB 873         ?ST=1  3
4186 122BE 3C          GOYES  POP#10         NOT REACHED LEVEL MARK YET
4187 122C0 03    POP#27 RTNCC
4188 122C2 8E3D  POP#30 GOSUBL STPASG
           7F
4189 122C8 8A4         ?A#B   A              REACHED TOP OF TABLE ?
4190 122CB AE          GOYES  POP#25         IF NOT, POP OFF THE LEVEL MARKER
4191 122CD 6E3E        GOTO   IODALj         DEALLOCATES THE ASSIGN TABLE
```

```
4192                *
4193                *
4194                ********************************************************************
4195                ********************************************************************
4196                **
4197                ** Name:    FNDMRK  -  Find Lowest Marker in Assign Table
4198                **
4199                ** Category:   FILUTL
4200                **
4201                ** Purpose:  Find the lowest level marker in the Assign table
4202                **
4203                ** Entry:  Nothing
4204                **
4205                ** Exit:  Carry set => Level marker found
4206                **             D1 points at the level marker
4207                **             C=DAT1  A
4208                **
4209                ** Calls:  I/OFND
4210                **
4211                ** Used:  A,C,D1
4212                **
4213                ** Stk lvls: +1
4214                ********************************************************************
4215                ********************************************************************
4216                *
4217 122D1 8E4D 5F =FNDMRK GOSUBL BUFASN
4218 122D7 500           RTNNC                 ASSIGN TABLE NOT FOUND
4219 122DA 137           CD1EX
4220 122DD CA            A=A+C  A
4221 122DF 131           D1=A                  D1 POINTS PAST END OF BUFFER
4222 122E2 1C4    FNDMK2 D1=D1- 5
4223 122E5 147           C=DAT1 A
4224 122E8 96E   =FNDMK- ?C#0   B
4225 122EB 7F            GOYES  FNDMK2         NOT REACH LEVEL MARKER YET
4226 122ED 02     FNDMK3 RTNSC
```

```
4227                    STITLE Find File# in File Info Buffer
4228                ****************************************************************
4229                ****************************************************************
4230                **
4231                ** Name:     FFIB#   -  Find File Number in FIB
4232                **
4233                ** Category:   FILUTL
4234                **
4235                ** Purpose:  Find File Number in File Information Buffer
4236                **
4237                ** Entry:    A(B) = File#
4238                **
4239                ** Exit:
4240                **           B  = File#
4241                **           P = 0
4242                **          Carry Clear:
4243                **              D1 @ File# in File Information Buffer
4244                **              B  = File#
4245                **          Carry Set:
4246                **              File# not found
4247                **
4248                **
4249                ** Calls:    I/OFND
4250                **
4251                ** Uses:     A,B,C,D1,P
4252                **
4253                **           I/OFND uses A,C,D1
4254                ** Stk lvls: +1
4255                **
4256                ** Detail:   B <-- File#
4257                **           C <-- FIB ID
4258                **           Find FIB Buffer  (I/OFND)
4259                **           If not found
4260                **              RTNSC
4261                **           Repeat until (File# = 00)
4262                **              Read File#
4263                **              If File# = Requested File#
4264                **                 RTNCC
4265                **              Skip to next file
4266                **           RTNSC
4267                ****************************************************************
4268                ****************************************************************
4269 122EF D8    =FFIB#  B=A     A               Save File#
4270 122F1 20            P=      0
4271 122F3 8E9A          GOSUBL  BUFFIB          Find File Information Buffer
            5F
4272 122F9 53F           GONC    FNDMK3          Not found, return set carry
4273 122FC D2            C=0     A
4274 122FE 31F3          LC(2)   =LFIB           Length of Info per File#
4275 12302 14B   FIB#10  A=DAT1  B               Read file#
4276 12305 968           ?A=0    B               End of buffer ?
4277 12308 00            RTNYES
4278 1230A 960           ?A=B    B               File# equal ?
4279 1230D 3B            GOYES   POP#27          If so, return carry clear
4280 1230F 133   FIB#20  AD1EX
```

```
4281 12312 CA          A=A+C  A            Skip to next file# info
4282 12314 133         AD1EX
4283 12317 6AEF        GOTO   FIB#10       Continue
4284            *
4285            *
4286 1231B             END
```

```
 A-MULT  Ext                  -  2017
 ARGERR  Ext                  -  1097
 ASN020  Abs   73544 #11F48 -  3614  3631  3639
 ASN050  Abs   73558 #11F56 -  3618  3612
 ASN055  Abs   73617 #11F91 -  3640  3636
 ASN060  Abs   73627 #11F9B -  3643  3619
 ASN070  Abs   73647 #11FAF -  3653
 ASN100  Abs   73663 #11FBF -  3659  3644
 ASN110  Abs   73695 #11FDF -  3671  3667
 ASN120  Abs   73761 #12021 -  3689  3660
 ASN130  Abs   73772 #1202C -  3692  3697  3699
 ASN150  Abs   73776 #12030 -  3693  3690
 ASN200  Abs   73780 #12034 -  3695  3657
 ASN300  Abs   73798 #12046 -  3701
 ASN80   Abs   73655 #11FB7 -  3656  3654
 ASNRTN  Abs   73552 #11F50 -  3616  3708
 ASNerr  Abs   73687 #11FD7 -  3668  3655
=ASSIGN  Abs   73523 #11F33 -  3607
 ASSNDC  Ext                  -  3605
 ASSNP   Ext                  -  3606
 AVMEMS  Abs  193940 #2F594 -    14  2801
 BADTYP  Abs   70767 #1146F -  1130   643
=BUFASN  Abs   71851 #118AB -  2270  1202  1262  2841  3857  4166  4217
=BUFFIB  Abs   71842 #118A2 -  2268  3050  3114  3847  3882  3919  3977  4026
                             4271
 BsErr   Ext                  -  4084
 CH#NFD  Abs   70826 #114AA -  1169  1203  1211  1215  1269
 CHKEOL  Ext                  -   384   388
=CHKRSP  Abs   74294 #12236 -  4081
 CHN#SV  Abs  194927 #2F96F -    14  1089  3711
=CLOSE#  Abs   73841 #12071 -  3746  3615
=CLOSEA  Abs   73956 #120E4 -  3847  3855
=CLOSEF  Abs   73863 #12087 -  3754  2863  4180
 CLSCH#  Abs   74179 #121C3 -  3993  3986
 CMPLER  Abs   70054 #111A6 -   443   436
 CREADC  Ext                  -  1633
=CREATE  Abs   71079 #115A7 -  1635
 CREATF  Ext                  -  3677
 CREATP  Ext                  -  1634
 CRETF+  Ext                  -  1974
=CRFSB-  Abs   71268 #11664 -  1743  1983  3680
 CRTERR  Abs   71258 #1165A -  1699  1639
=CRTF    Abs   71361 #116C1 -  1879
 CRTF-   Abs   71377 #116D1 -  1887  1696
 CRTF05  Abs   71396 #116E4 -  1894  1889
 CRTF10  Abs   71422 #116FE -  1908  1905  1907
 CRTF20  Abs   71438 #1170E -  1915  1912
 CRTF25  Abs   71446 #11716 -  1918  1916
 CRTF30  Abs   71464 #11728 -  1927  1897
 CRTF35  Abs   71478 #11736 -  1933  1920
 CRTF40  Abs   71483 #1173B -  1936  1895
 CRTF45  Abs   71501 #1174D -  1942  1892
 CRTF46  Abs   71513 #11759 -  1946  1944
=CRTF50  Abs   71517 #1175D -  1952  1939
 CRTF70  Abs   71626 #117CA -  2008  1946
```

```
 CRTFer  Abs   71456 #11720 -  1922  1909  1928  1934  2850
 CTE100  Abs   71103 #115BF -  1645  1637
 CTE110  Abs   71181 #1160D -  1673  1666  1669
 CTE120  Abs   71229 #1163D -  1690  1686
 CTE300  Abs   71262 #1165E -  1701   580  1697  3616
 Cslc5   Ext                -  1963  3640
 D0=FIB  Ext                -  1010  1047
 D0=PCA  Ext                -  1678
 D1=AVE  Ext                -  3626
 D1FSTK  Ext                -  1086
 DATL10  Abs   73444 #11EE4 -  3559  3504  3513
 DATLN2  Abs   73317 #11E65 -  3506  3499
 DATLN4  Abs   73341 #11E7D -  3517  3507
 DATLN6  Abs   73354 #11E8A -  3525  3518
 DATLN8  Abs   73417 #11EC9 -  3549  3526
 DATLN9  Abs   73442 #11EE2 -  3558  3523  3541  3547
 DATSTR  Abs   72801 #11C61 -  3152
 DELASN  Abs   73987 #12103 -  3857  3848  3883  3886
 DELBF5  Abs   73898 #120AA -  3805
 DELBF8  Abs   73931 #120CB -  3816  3801  3812
=DELFIB  Abs   73877 #12095 -  3796  3407  3854  3888  4002
=DLFIBA  Abs   74000 #12110 -  3882  3889
 DTLNER  Abs   73337 #11E79 -  3514  3511
=ENDFIL  Abs   70263 #11277 -   664   698
 ERREX   Abs   71097 #115B9 -  1643   665  1129  1683
 ERRTN   Abs   72496 #11B30 -  3034  3028
=EXPCH#  Abs   72301 #11A6D -  2791  2866
 EXPEX-  Ext                -   387   595
 EXPR    Ext                -   391  1087
 EXTCHK  Abs   70957 #1152D -  1309  1665
=F#NFND  Abs   70759 #11467 -  1128  1132  1134
 FASC10  Abs   69876 #110F4 -   304   296   300   302
 FASC20  Abs   69887 #110FF -   331   349
 FASC25  Abs   69896 #11108 -   334   181
 FASC30  Abs   69898 #1110A -   335   333
 FASC40  Abs   69939 #11133 -   350   339
=FASCFD  Abs   69827 #110C3 -   289
=FASCH   Abs   69884 #110FC -   330   295
=FDCH#   Abs   70828 #114AC -  1202  1126  2855  3702  3746
=FDCH#-  Abs   70835 #114B3 -  1205
 FDCH40  Abs   70846 #114BE -  1210  1217  1229
 FDCH50  Abs   70879 #114DF -  1223  1228
 FDCH60  Abs   70896 #114F0 -  1232  1219  1220  1275
 FDENT1  Abs   72557 #11B6D -  3060  3071
 FDF#10  Abs   70917 #11505 -  1267  1272  1274  1277
 FDF#20  Abs   70920 #11508 -  1268  1265
=FDFIL#  Abs   70904 #114F8 -  1262  3996
=FFIB#   Abs   74479 #122EF -  4269  1133  3754
 FIB#10  Abs   74498 #12302 -  4275  4283
 FIB#20  Abs   74511 #1230F -  4280
 FIBA10  Abs   70771 #11473 -  1131  1127
=FIBAD-  Abs   70776 #11478 -  1133
=FIBADR  Abs   70743 #11457 -  1123   583
 FIBO/F  Abs   74037 #12135 -  3919  3916
=FIBOFF  Abs   74034 #12132 -  3917
```

```
=FIBON   Abs   74027 #1212B -  3915
 FILEP!  Ext                 -  291
 FILOPN  Abs   72592 #11B90 -  3073  3067  3084  3110
 FINDF   Ext                 -  3036
 FLHA10  Abs   74220 #121EC -  4029  4037
 FLHA20  Abs   74235 #121FB -  4034
=FLSHIO  Abs   74247 #12207 -  4065  3758  3853  4033
=FLUSHA  Abs   74208 #121E0 -  4026
 FNDFBF  Ext                 -  711
=FNDMK-  Abs   74472 #122E8 -  4224
 FNDMK2  Abs   74466 #122E2 -  4222  4225
 FNDMK3  Abs   74477 #122ED -  4226  4272
=FNDMRK  Abs   74449 #122D1 -  4217  4154
 FPOLL   Ext                 -  2875
 FSPECx  Ext                 -  2016
=FTBSCH  Abs   69779 #11093 -  176   128   186
 FTSH20  Abs   69805 #110AD -  185   191
 FTYP05  Abs   69767 #11087 -  139   136
 FTYP10  Abs   69769 #11089 -  141   129   135
 FTYPE   Ext                 -  127   294
=FTYPER  Abs   71093 #115B5 -  1641  1130  1688
=FTYPF#  Abs   69721 #11059 -  125   1378  3404
=FTYPFD  Abs   69715 #11053 -  123   3156
 GARG10  Abs   69993 #11169 -  394   403
 GARG30  Abs   70005 #11175 -  398   406
=GETARG  Abs   69950 #1113E -  384   1647
=GETBUF  Abs   74311 #12247 -  4121  3391
=GETC#-  Abs   70702 #1142E -  1087
=GETCH#  Abs   70695 #11427 -  1086  570   3607
=GETRE#  Abs   70086 #111C6 -  583   571
 GF#130  Abs   72602 #11B9A -  3077  3062
 GF#150  Abs   72617 #11BA9 -  3083  3093
 GF#170  Abs   72647 #11BC7 -  3095  3087  3089
 GF#180  Abs   72697 #11BF9 -  3114  3103
 GTBF25  Abs   74339 #12263 -  4129  4082
 GTBF30  Abs   74341 #12265 -  4130  4127
=GTSVC#  Abs   73827 #12063 -  3711  2854  2868  3614  3695  3701
 H41#10  Abs   70344 #112C8 -  699   733
 HP41R#  Abs   70316 #112AC -  687   650
=I/OAL+  Abs   72059 #1197B -  2549
=I/OALL  Abs   72061 #1197D -  2551  2845  4126
=I/OCOL  Abs   72057 #11979 -  2547
=I/OCON  Abs   71968 #11920 -  2404  4184
 I/OCRE  Abs   72172 #119EC -  2606  2555
=I/ODAL  Abs   72257 #11A41 -  2771  3802  3815  3859
=I/OEX2  Abs   72207 #11A0F -  2674
=I/OEXP  Abs   72209 #11A11 -  2675  2792  3102
 I/OFN+  Abs   71860 #118B4 -  2273  2551  2675
=I/OFND  Abs   71866 #118BA -  2276  2211  2269  2271  2337  2406  2773
=I/ORES  Abs   71935 #118FF -  2333  3937  3945
 I/ORT+  Abs   72164 #119E4 -  2601  2573
 I/ORTN  Abs   72170 #119EA -  2604  2679
 I/Obuf  Abs      10 #0000A -  566   610   613   707   753   798   801  1016
                             3384  3390  3451  3468  3484
 IDIV    Ext                 -  2146
```

```
 IDLNSV  Abs   72234 #11A2A -  2727  2411  2553  2680
 INIT10  Abs   71713 #11821 -  2111  2106  2115
 INIT20  Abs   71715 #11823 -  2114  2097
 INIT30  Abs   71745 #11841 -  2132
 INIT70  Abs   71802 #1187A -  2163  2120
 INIT75  Abs   71808 #11880 -  2165  2123
=INITMF  Abs   71661 #117ED -  2087  2003
 IOBFEN  Abs  193910 #2F576 -    14  2610  2801  2802
 IOCN10  Abs   72025 #11959 -  2425  2418
 IOCN20  Abs   72027 #1195B -  2426  2424
=IOCNDO  Abs   71973 #11925 -  2406  3822
 IODAL2  Abs   72271 #11A4F -  2776  2772
 IODALj  Abs   73996 #1210C -  3859  4191
 IOEX24  Abs   72079 #1198F -  2562  2682
 IOEX25  Abs   72089 #11999 -  2566  2617
 IOEX40  Abs   72139 #119CB -  2591  2544
=IOFNDO  Abs   71873 #118C1 -  2279  2336  2771
 IOFND3  Abs   71890 #118D2 -  2283  2297
 IOFND5  Abs   71914 #118EA -  2292  2282
 IOFSC5  Abs   71827 #11893 -  2211  2214
=IOFSCR  Abs   71822 #1188E -  2210  4121
 IONULL  Abs   72055 #11977 -  2546  2410
 IOSH10  Abs   72031 #1195F -  2536  2558
 IOSH20  Abs   72048 #11970 -  2543  2426
 LIF#05  Abs   70382 #112EE -   716   708
 LIF#07  Abs   70406 #11306 -   725   761
 LIF#10  Abs   70409 #11309 -   726   756
 LIF#30  Abs   70417 #11311 -   730   738   740
 LIF#40  Abs   70427 #1131B -   735   728
 LIF#50  Abs   70469 #11345 -   750   748
 LIF#60  Abs   70490 #1135A -   758   754
 LIF1R#  Abs   70352 #112D0 -   706   648
 MAINEN  Abs  193905 #2F571 -    14  2279
 MEMCL+  Ext                -  2572
 MOVED2  Ext                -  2586
 MOVEUA  Ext                -  2800
 MOved3  Ext                -  3125
 MfErr   Ext                -  1643
 NEXTst  Ext                -  1701
 NOARG   Abs   70021 #11185 -   404   385
 NOMEM   Ext                -  3112
 NOROOM  Abs   72387 #11AC3 -  2850  2846  2867  3393
 NXTFIL  Abs   74124 #1218C -  3948  3930  3936
 ONEARG  Abs   70015 #1117F -   402   389
=OPENF   Abs   72454 #11B06 -  3017
=OPENF*  Abs   72465 #11B11 -  3021  3018
=OPENF-  Abs   72532 #11B54 -  3048  3031  3698
 OPN120  Abs   72393 #11AC9 -  2854  2842
 OPN150  Abs   72425 #11AE9 -  2866  2856
 OPN200  Abs   72439 #11AF7 -  2871  2858
=OPNCH#  Abs   72361 #11AA9 -  2841  2849  2864  3696
 OPNERR  Abs   72461 #11B0D -  3019  1699
=OPNF+   Abs   72511 #11B3F -  3041
 OPNF+1  Abs   72769 #11C41 -  3140  3041  3689
 OPNF+2  Abs   72850 #11C92 -  3170  3045  3693
```

```
 OPNF+3  Abs   72898 #11CC2 -  3185  3029  3656
 OPNF20  Abs   72504 #11B38 -  3038  3022
 OPNF25  Abs   72524 #11B4C -  3045  3042
 OPNF50  Abs   72532 #11B54 -  3050  3046
 OPNOFD  Abs   72490 #11B2A -  3033  3039  3668
 OUTRNG  Abs   70737 #11451 -  1097   441  1094
 POLL    Ext               -  1771
 POLLFF  Abs   71331 #116A3 -  1765  3026  3649
 POP#10  Abs   74369 #12281 -  4166  4159  4186
 POP#20  Abs   74415 #122AF -  4182  4177
 POP#25  Abs   74421 #122B5 -  4184  4190
 POP#27  Abs   74432 #122C0 -  4187  4279
 POP#30  Abs   74434 #122C2 -  4188  4174
 POP1N   Ext               -   434
=POPARG  Abs   70030 #1118E -   434   392   397   596  1088
=POPCH#  Abs   74343 #12267 -  4154
 PTRAD2  Ext               -  2803
 PUFB10  Abs   74142 #1219E -  3979  3991
 PUFB20  Abs   74200 #121D8 -  4000  3997
=PUGFIB  Abs   74136 #12198 -  3977
 RCLM10  Abs   74049 #12141 -  3922  3951
 RCLM20  Abs   74090 #1216A -  3937  3932
 RCLM30  Abs   74121 #12189 -  3946  3942
 RDATTY  Ext               -   443
 RDLNAS  Ext               -   735
 RES#10  Abs   70082 #111C2 -   580   573
=RESTDO  Abs   70301 #1129D -   679
=RESTR#  Abs   70060 #111AC -   569
=REWIND  Abs   70501 #11365 -   792  3584
 RNMERR  Ext               -  1671
 RTNSCj  Abs   71966 #1191E -  2345  2288
 RWND10  Abs   70542 #1138E -   805   800
 Random  Abs       9 #00009 -   567   572   584   598
 S-R0-0  Abs  194673 #2F871 -    14  1314  1745
 S-R1-0  Abs  194689 #2F881 -    14  1954  2008
 S-R1-2  Abs  194699 #2F88B -    14  1383
 SAVDO+  Abs   70807 #11497 -  1164
=SAVDO-  Abs   70813 #1149D -  1166
 SAVDO1  Abs   70823 #114A7 -  1168  1140
=SAVEDO  Abs   70810 #1149A -  1165   602
 SCRTCH  Abs  194817 #2F901 -    14  3191  3491  3528
 SFPT10  Abs   70596 #113C4 -  1018  1033
 SFPT15  Abs   70601 #113C9 -  1020
 SFPT20  Abs   70603 #113CB -  1022  1017
 SFPT40  Abs   70661 #11405 -  1047  1040
 SKPBYT  Ext               -   755
 SNAPSV  Ext               -  1654  2874
 SRE#30  Abs   70195 #11233 -   633   612   614
 SRE#35  Abs   70222 #1124E -   643   628   631
 SRE#40  Abs   70226 #11252 -   645   642
 SRE#41  Abs   70238 #1125E -   649
 SRE#PL  Abs   70178 #11222 -   626   646
 SREC#A  Abs   70130 #111F2 -   600   578
=STFPTR  Abs   70564 #113A4 -  1007   675   710   805
 STMTDO  Abs  194705 #2F891 -    14   679  1166
```

```
 STMTD1  Abs  194710 #2F896 -    14   603   792   806  1136  4065
 STMTR0  Abs  194673 #2F871 -    14  1656  1765  3057  3140  3170  3185  3380
                            3395  3464  3476
 STMTR1  Abs  194689 #2F881 -    14  1690  3664
=STPASG  Abs   72347 #11A9B -  2805  2791  2844  4182  4188
 SVFSP+  Abs   70953 #11529 -  1308  1645  3021  3643
 SVFSPC  Abs   70968 #11538 -  1314  1308  1879
 SVFT90  Abs   71061 #11595 -  1394  1379  1945  3043
=SVFTYP  Abs   71008 #11560 -  1378  1682  1881
 Snaprs  Ext                -  1673  3474
 TIT#10  Abs   70253 #1126D -   659
 TIT#20  Abs   70271 #1127F -   667   660   662
 TIT#30  Abs   70291 #11293 -   673   701
 TYPER   Abs   72518 #11B46 -  3043  3408  3691
 WIPOUT  Ext                -  2095  3131
 WRTF10  Abs   72991 #11D1F -  3395  3392
 WRTF20  Abs   73011 #11D33 -  3401  3386
 WRTF25  Abs   73040 #11D50 -  3410  3405
 WRTF30  Abs   73154 #11DC2 -  3448  3444
 WRTF40  Abs   73197 #11DED -  3464  3452
 WRTF41  Abs   73233 #11E11 -  3476
 WRTF42  Abs   73243 #11E1B -  3478  3469
 WRTF45  Abs   73272 #11E38 -  3490  3485
 WRTF50  Abs   73452 #11EEC -  3565  3486
 WRTF60  Abs   73472 #11F00 -  3572  3561
 WRTFER  Abs   73036 #11D4C -  3408  3514
=WRTFIB  Abs   72942 #11CEE -  3377  3136
=a-mult  Abs   71654 #117E6 -  2017   669  1919  3556
 bASSGN  Abs    2052 #00804 -    13  2270  2808
 bFIB    Abs    2051 #00803 -    13  2268  3101  3821
 bSCRTC  Abs    3584 #00E00 -    13  2210
 bserr   Abs   74305 #12241 -  4084  3019  3669  3692
 bserr1  Abs   73691 #11FDB -  3669  3624  3641  3651  3678
 chkspc  Ext                -  3109
 csrc5   Ext                -  1978  3625
 eCHNL#  Ext                -  1128
 eDVCNF  Ext                -  4083
 eEOFIL  Ext                -   664
 eFOPEN  Ext                -  3074
 eFTYPE  Ext                -  1394  1641
 eFnFND  Ext                -  3033
 eMEM    Ext                -  1922
 fBASIC  Abs   57876 #0E214 -    13  2104
=findf   Abs   72498 #11B32 -  3036  1668  3038  3659
 fltdh   Ext                -   437
 fpoll+  Abs   72441 #11AF9 -  2874  1042  1054  3471  4077
=fspecx  Abs   71648 #117E0 -  2016  1636  3017  3618
 lFIB    Abs      63 #0003F -    13  3070  3082  3108  3130  3819  3950  3990
                            4035  4274
 lFLENh  Abs       5 #00005 -    13  2089
=nomem   Abs   72691 #11BF3 -  3112
 oACCSb  Abs      11 #0000B -    13  1037  4072
 oCOPYb  Abs      10 #0000A -    13   606   608  1012  1014  3425  3429
 oCPOSb  Abs      40 #00028 -    13   672   673   699  1015  1050  3578  3948
                            3949
```

oDBEGb	Abs	21	#00015	-	13	634	655	687	716	1014	1015	3064
					3068	3465	3478	3808	3928	3948		
oDEVCb	Abs	12	#0000C	-	13	608	3429	3434				
oDLENb	Abs	46	#0002E	-	13	688	699	717	3483	3490	3497	3501
					3503	3551	3555	3574				
oFBEGb	Abs	13	#0000D	-	13	802	803	1048	1049	3063	3064	3434
					3449	3807	3808	3927	3928	3982	3984	
oFBF#b	Abs	2	#00002	-	13	795	802	3383	3398	3401	3798	3925
					3927	4068	4072					
oFIL#b	Abs	0	#00000	-	13	3063	3383	3398				
oFLAGh	Abs	20	#00014	-	13	3455						
oFLENh	Abs	32	#00020	-	13	1968	3163	3566				
oFSIZb	Abs	57	#00039	-	13	3490	3497					
oFTYPb	Abs	5	#00005	-	13	3401	3421	3462				
oFTYPh	Abs	16	#00010	-	13	3155						
oIMPLh	Abs	37	#00025	-	13	3177	3675					
oPROTb	Abs	9	#00009	-	13	3421	3425	3462	3544			
oREC#b	Abs	32	#00020	-	13	655	667	1049	1050	3478	3483	3501
					3503	3551						
oRECLb	Abs	36	#00024	-	13	667	672	687	688	716	717	3068
					3069	3555	3574	3576				
oRLENb	Abs	52	#00034	-	13	673	3069	3070	3576	3578	3949	3950
oSHLNb	Abs	19	#00013	-	13	803	3449	3465				
pCREAT	Abs	9	#00009	-	13	1941						
pCRT=8	Abs	35	#00023	-	13	1891						
pDATLN	Abs	42	#0002A	-	13	3510						
pDIDST	Abs	10	#0000A	-	13	3472						
pFASCH	Abs	44	#0002C	-	13	299						
pFINDF	Abs	23	#00017	-	13	3027	3650					
pFTYPE	Abs	45	#0002D	-	13	131						
pRDCBF	Abs	24	#00018	-	13	1055						
pSREC#	Abs	40	#00028	-	13	627						
pWRCBF	Abs	26	#0001A	-	13	1043	4078					
poll	Abs	71355	#116BB	-	1771	130	298	626	1890	1940	3509	
ptrad2	Abs	72331	#11A8B	-	2801	2580						
ptradj	Abs	72324	#11A84	-	2800	2543	2785					
stuff	Ext			-	2165							
t*	Ext			-	3610							
tCOMMA	Ext			-	589							
tEOL	Ext			-	586	2107						

Input Parameters

  Source file name is SC&FIL::MS

  Listing file name is SC/FIL:TI:ML::-1

  Object file name is SC%FIL:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *
 2            *       J  PPPP    &     PPPP   00000  L
 3            *       J  P   P  & &    P   P  0   0  L
 4            *       J  P   P  & &    P   P  0   0  L
 5            *       J  PPPP    &     PPPP   0   0  L
 6            *       J  P      & & &  P      0   0  L
 7            *    J  J  P      &  &   P      0   0  L
 8            *     JJJ  P       && &  P      00000  LLLLL
 9            *
10                    TITLE  POLL Routine <831212.1204>
11 1231B              ABS    #1231B
12                    RDSYMB TI%EQU::MS
13                    RDSYMB SB%RAM::MS
14           ********************************************************************
15           ********************************************************************
16           **
17           ** Name:(S) POLL    -  Poll LEX Files with Process Number
18           ** Name:(S) POLLD+  -  Poll LEX Files adjusting AVMEME in D(A)
19           **
20           ** Category:   POLL
21           **
22           ** Purpose:
23           **      Poll All LEX Files for Special Processing
24           **      Pass a Process #, and Parameters to each LEX File
25           **
26           **      POLLD+ entry used for routines needing to pass AVMEME
27           **             in register D(A).  This value is adjusted to
28           **             reflect the save area used by POLL.
29           **             Currently used by Parse and Decompile.
30           **
31           ** Entry:
32           **      POLLD+: Sets D(A) to what AVMEME will be during poll,
33           **                then falls through to POLL
34           **              Used during Parse and Decompile
35           **
36           **              Example:
37           **
38           **              GOSBVL =AVS=DO      Set AVMEMS @ DO
39           **              GOSBVL =POLL        Issue Poll
40           **              CON(2) =pDEVCp      Device Parse Poll
41           **              GOSBVL =D=AVME      Reset D(A) @ AVMEME
42           **              GOC    ErrRtn       Error Return
43           **              ?XM=0
44           **              GOYES  Handle       Handled by LEX File
45           **              ...                 Not handled
46           **
47           **      POLL:   Process# @ Calling Routine Return Address
48           **              Process# = CON(2)
49           **              HEX Mode
50           **
51           **              Example:
52           **
53           **              GOSBVL =POLL        Issue Poll
54           **              CON(2) =pFILXQ      File execute poll
55           **              GOC    ErrRtn       Error Return
```

```
56          **              ?XM=0
57          **              GOYES  Handle     Handled by LEX file
58          **              ...               Not handled
59          **
60          **      Assumes:
61          **        MTHSTK is active when called
62          **          Any routine polling with active stack must update
63          **          AVMEME to top of stack pointer
64          **        Uses SNAPBF for temporary storage of registers
65          **        Uses SAVSTK to stack POLL information
66          **        Moves memory from FORSTK --> AVMEME before Save
67          **        Memory check w/o LEEWAY for Poll Save Area
68          **        Calling routine return address saved on GOSUB stack
69          **          Will be updated if memory moves during poll
70          **
71          **      Entry to LEX File Poll Routine:
72          **        Carry Clear to indicate Normal Poll (see FPOLL)
73          **        B(A) = Process#
74          **               B(B)   = Process #
75          **               B(2-4) = 0
76          **        A,D,D0,D1 = Original Contents from Calling Routine
77          **        R registers, Status are untouched by POLL
78          **        R0-R3, status cannot be destroyed while identifying
79          **          Process#.
80          **        3 levels of subroutine stack saved
81          **        One more stack level available than routine issuing
82          **          the Poll
83          **
84          ** Exit:
85          **      Carry set
86          **        Insufficient Memory to Issue Poll OR
87          **        Error return / "Something Funny" from LEX File
88          **        All registers & pointers preserved from LEX File
89          **            EXCEPT A,B
90          **        A,C have the same value on return
91          **            The contents of C on return from LEX file are
92          **            saved in A, then put back into C before return
93          **            Allows LEX Files to return Error # in C(0-3)
94          **
95          **        If not enough memory to save POLL info
96          **            C(0-3) <-- eMEM
97          **
98          **        A routine issuing POLL should check for CARRY
99          **          If there was not enough memory to issue the poll
100         **          this exception should be noted/indicated.
101         **
102         **      Carry clear
103         **        Look @ XM to determine if handled
104         **        If XM=0
105         **           Process has been handled by LEX File
106         **           All regs & ptrs preserved from LEX File
107         **               EXCEPT B,C
108         **           A is NOT destroyed
109         **
110         **        If XM=1
```

```
111             **          Process has NOT been handled
112             **          Registers & pointers restored to Entry values
113             **              EXCEPT B,C
114             **          A is not destroyed
115             **
116             **       A POLL responder must return with CARRY CLEAR
117             **          if NOT error return or NOT handled
118             **
119             **       A POLL responder must return with CARRY SET
120             **          if Error return
121             **          C(0-3) should hold the Error Number
122             **             It will be saved in A, then restored to C
123             **
124             **     Take over handler
125             **       The poll responder does not return to the POLL routine
126             **       This is allowed by certain individual polls as
127             **           indicated in their documentation headers
128             **       The handler MUST:
129             **           GOSBVL =COLLAP  to collapse POLL Save info
130             **           GOSBVL =POPUPD  to pop POLL issuer's rtn address
131             **
132             ** Calls:      SALLOC, CRGJMP, FIRSAV, RESRTN, RESSVA,
133             **
134             **             GLXPOL, SNAPSV, SNAPRS, MOVEU3, SNAPBF
135             **             RSTK<R, PSHSTL, MEMCKL
136             **
137             ** Uses:
138             **  Exclusive: B,C,SNAPBF,P, SAVSTK, 2 levels in RSTKBF
139             **  Inclusive: B,C,SNAPBF,P, SAVSTK, 2 levels in RSTKBF
140             **
141             ** Stk lvls:   Preserves all levels
142             **             POLL saves Calling Return Address on GOSUB
143             **             stack so it will be updated if memory moves
144             **
145             ** Detail:
146             **           B(A)      = Process#
147             **           B(B)      = Process#
148             **           B(2-4)    = 0
149             **
150             **           Save Stack:
151             **                                                  Low Memory
152             **
153             ** =lAp      A                                  16
154             ** =lDp      D                                  16
155             ** =lD1p     D1                                  5
156             ** =lD0p     D0                                  5
157             ** =lPOL#p   Poll Number                         5
158             ** =lRTN2p   Rtn Level 2                        5
159             ** =lRTN3p   Rtn Level 3                        5
160             ** =lBPOSp   Relative Position in LEX Buffer     5 @ SAVSTK - 5
161             **                                                  High Memory
162             **           Length of Save Stack =             62 = 3E hex
163             **
164             **           GOSUB Stack:
165             **
```

```
166             **            ---------------
167             **  GSBSTK -> |F| Rtn Addr 1 |                 6
168             **            |--------------|
169             **
170             **          Return Type = F indicate an Update Address
171             **
172             **      Original contents of A,D,D0,D1 sent to each LEX File
173             **      Carry clear, XM=0 on call to each LEX File
174             **      R registers and status untouched
175             **
176             **      3 levels of subroutines saved
177             **        Return Level 1 is saved on GOSUB stack as update
178             **          address incase memory moves during a Poll
179             **        Return Level 2 and 3 are saved in SAVSTK area
180             **      Information saved by POLL is stacked from SAVSTK
181             **        toward LOW memory.
182             **          AVMEME is adjusted above saved information.
183             **          AVMEME is readjusted when POLL returns
184             **      This allows POLL to be called "recursively"
185             **
186             **      If "Error" Response from LEX File  (Carry Set)
187             **         All registers & pointers left intact EXCEPT A,B
188             **         A is used to save C during restore
189             **         A,C have the returned value of C when rtn to Caller
190             **         C(0-3) <-- Error number
191             **
192             **      If "normal" Response from LEX File (XM=0)
193             **         All registers & pointers left intact EXCEPT C,B
194             **         C has the value of A on return
195             **
196             **      If no LEX File respond (XM=1)
197             **         Restore A,D,D0,D1 to entry values
198             **         RTNSXM to Calling Routine
199             **         C is NOT set to A
200             **
201             **      If LEX File wishes Poll to continue to others
202             **         Carry must be clear, XM=1
203             **         If "non-original" contents of A,D,D0,D1 are
204             **         to be sent to other LEX Files, the information
205             **         above SAVSTK can be altered by the LEX File
206             **
207             ** Algorithm:
208             **
209             **      Save A,D,D0,D1,Rtn Lvl 1 temporary in SNAPBF (SNAPSV)
210             **      If not enough memory to save info           (SALLOC)
211             **         Restore saved registers and pointers     (SNAPRS)
212             **         Adjust return address past Process #
213             **         C <--- Error Number                      (eMEM)
214             **         RTNSC
215             **      Save Return Level 2 & 3 cus SALLOC uses 2   (D,D0)
216             **      Allocate SAVSTK area                        (SALLOC)
217             **      Restore Rtn Lvl 3,2 to stack
218             **      Move temporary save info to SAVSTK          (MOVEU3)
219             **      Read Return Level 1, read process# @ Rtn address
220             **      Write process # over Return Level 1 location
```

```
221             **      Adjust Return Level 1 past Process #, saving in A(A)
222             **      Save Rtn Levels 2,3 in SAVSTK
223             **      Save 2 levels in RSTK Buffer                    (R<RSTK)
224             **      Push Rtn Level1 on GOSUB stack to be updated (PSHSTL)
225             **        Make sure LEEWAY is NOT checked when pushing
226             **      Restore Return Levels                           (RSTK<R)
227             **      Initialize Relative Offset into LEX Buffer to 0
228             ** 1:   Get LEX File POLL address                       (GLXPOL)
229             **      If LEX file POLL address                 (Carry Clear)
230             **         Save updated Rel. Position into Buffer in SAVSTK
231             **         Push LEX file jump address on stack
232             **         Restore registers,pointers,process#          (RESSVA)
233             **         Pop LEX file jump address
234             **         Clear XM flag
235             **         Gosub to LEX File Poll Routine               (CRGJMP)
236             **         Clear B(S) to save carry from LEX file return
237             **         If Carry set  (Error response from LEX File)
238             **            Set A=C to preserve C during restore
239             **            goto 2;
240             **         If LEX File responded                        (XM=0)
241             **            Set B(S) = 1
242             **            goto 2;
243             **         else       (No response)                     (XM=1)
244             **            Restore Relative Position in LEX Buffer
245             **            goto 1; (Continue polling)
246             **    2:   Save current A,D,D0,D1 in SNAPBF and
247             **         Restore return lvls 2,3; Release SAVSTK     (RESRTN)
248             **         Restore current A,D,D0,D1,Rtn Lvl 1         (SNAPRS)
249             **         Push Rtn Lvl 1 back on stack
250             **         Set C=A
251             **         Return indicating carry from LEX File       (B(S))
252             **   else
253             **        (No more LEX Files in LEXBUF)
254             **      Restore A,D,D0,D1 from SAVSTK                  (RESSVA)
255             **      Save A->D1,Restore Rtn Lvls, Release SAVSTK    (RESRTN)
256             **      Restore A->D1,Rtn1 from SNAPBF                 (SNAPRS)
257             **      Push Rtn Lvl 1 back on stack
258             **      RTNSXM
259             **
260             ** History:
261             **
262             **    Date     Programmer          Modification
263             **   --------  ----------  ----------------------------------
264             **  07/13/82   JP          Modified documentation
265             **  10/14/82   JP          No Leeway check when allocate Save Ar
266             **  10/14/82   JP          Rewrote to interface to SNAPBF
267             **  01/31/83   JP          After SALLOC, restore RSTK from D0
268             **  02/05/83   JP          Rewrote to save Rtn Addrs on GSBSTK
269             **  02/05/83   JP          Set XM=1 if Carry set/Error return
270             **  02/15/83   JP          No Leeway Check when PSHSTK called
271             **  03/01/83   JP          Added lPOLra to D(A) in POLLD+ entry
272             **  06/01/83   JP          Added MEMCHK of (lPOLSV + lRTNADR)
273             **
274             ********************************************************************
275             ********************************************************************
```

```
276             *
277             * Not enough memory to save info
278             *   Restore Registers, Pointers
279             *   Adjust Return address past Process#
280             *   Error return
281             *
282 1231B 7031  POLERR GOSUB  Snaprs         Restore registers, pointers
283 1231F E6          C=C+1  A              Move past Process#
284 12321 E6          C=C+1  A
285 12323 06          RSTK=C
286 12325 3300 =RMEM  LC(4)  =eMEM          Insufficient Memory
          00
287 1232B 02          RTNSC                 Set carry for Error Return
288             *
289             * Set D(A) to new AVMEME
290             *   Subtract:  Poll Save Area + Return Addr on GOSUB Stack
291             *
292 1232D D2  =POLLD+ C=0    A
293 1232F 20          P=     0
294 12331 3144        LC(2)  (=lPOLSV)+(lPOLra)
295 12335 E3          D=D-C  A
296             *
297             * Pop Return Level 1
298             * Save A,D,D0,D1,Rtn Level 1 in SNAPBF
299             *
300 12337 07  =POLL   C=RSTK                Rtn Lvl 1
301 12339 8F00        GOSBVL =SNAPSV        Save A,D,D0,D1,Rtn Lvl 1
          000
302             *
303             * Check memory w/o LEEWAY
304             *   Check for POLL save AND Calling return address on GOSUB stack
305             *   Avoids eMEM problem later with PSHSTL
306             *
307 12340 20          P=     0
308 12342 D2          C=0    A
309 12344 3144        LC(2)  (lPOLSV)+6     POLL save area + Rtn Addr on stk
310 12348 21          P=     1              No LEEWAY check
311 1234A 8F00        GOSBVL =MEMCKL        Check memory
          000
312 12351 49C         GOC    POLERR         Insufficient memory
313             *
314             * P=0 from MEMCKL
315             * Save Rtn Lvl 2 & 3 because SALLOC uses 2
316             * Allocate Save Area
317             *   On return from SALLOC
318             *      D0 holds Return Level 3
319             * No need to check carry from SALLOC due to prior MEMCKL call
320             * Restore Return Level 3,2
321             *
322 12354 07          C=RSTK                Rtn Lvl 2
323 12356 D7          D=C    A              Save Rtn Lvl 2
324 12358 07          C=RSTK                Rtn Lvl 3
325 1235A 134         D0=C                  D0 preserved during SALLOC
326 1235D D2          C=0    A              P=0 from MEMCKL
327 1235F 31E3        LC(2)  lPOLSV         Length of POLL Save Information
```

```
328 12363 21          P=     1              No Leeway Check for Save allocation
329 12365 8F00        GOSBVL =SALLOC        Memory is sufficient
          000
330 1236C 136         CDOEX                 Return Level 3
331 1236F 06          RSTK=C                Restore Rtn Lvl 3
332 12371 DB          C=D    A              Return Level 2
333 12373 06          RSTK=C                Restore Rtn Lvl 2
334             *
335             * Move temp save infor (SNAPBF) to SAVSTK
336             *   On exit from SALLOC:
337             *      D1   @ Start of Destination = (SAVSTK) - lPOLSV
338             *   Set:
339             *      D0   @ Start of Source        SNAPBF
340             *      C(A) = Block Length           47 nibs (SNAPBF length)
341             *
342 12375 1B0F          D0=(5) =SNAPBF        Start of Source
          7F2
343 1237C D2          C=0    A
344 1237E 31F2        LCHEX  2F             Length of SNAPBF  (47 nibs)
345 12382 8E00        GOSUBL =Moveu3        Move A,D,D0,D1,Rtn1 to SAVSTK
          00
346             *
347             * On exit from MOVEU3:
348             *    D1 @ End of Destination  = Past Rtn1 address
349             *
350             * Read Poll# @ Return Level 1
351             * Write Poll# into Save Area, on top of Rtn1 address
352             * Adjust Rtn Addr 1 past Poll#, save in A
353             *
354 12388 1C4           D1=D1- lRTN1p         Position @ Rtn addr 1
355 1238B 143         A=DAT1 A              Read Rtn addr 1
356 1238E 130         D0=A
357 12391 D2          C=0    A
358 12393 14E         C=DAT0 B              Read Poll # @ Rtn addr 1
359 12396 145         DAT1=C A              Write Poll# over Rtn1 address
360 12399 E4          A=A+1  A
361 1239B E4          A=A+1  A              Adjust Rtn addr 1 past Poll#
362             *
363             * Save Rtn Level 2,3 in SAVSTK area
364             * Save 2 stack levels before PSHUPD call
365             *   PSHUPD uses 3, POLL needs to save 4 (2 already saved)
366             *   A(A) = Address to Push
367             * Push Rtn Addr 1 on GOSUB stack to be updated
368             * Restore 2 stack levels
369             *
370 1239D 174           D1=D1+ lPOL#p         Move past Poll#
371 123A0 07          C=RSTK
372 123A2 145         DAT1=C A              Save Rtn Level 2
373 123A5 174         D1=D1+ lRTN2p         Move past Rtn Level 2
374 123A8 07          C=RSTK
375 123AA 145         DAT1=C A              Save Rtn Level 3
376 123AD 7FA0        GOSUB  R<rstk         Save 2 stack levels
377             *
378             * Duplicate PSHUPD code so LEEWAY is NOT checked
379             *
```

```
380                *
381 123B1 AC3              D=0    S              Return type nibble
382 123B4 A4F              D=D-1  S              Set=Update address
383 123B7 D6               C=A    A              Save Return address
384 123B9 06               RSTK=C                on Hardware stack
385 123BB D2               C=0    A
386 123BD 306              LCHEX  6              # nibbles to open up
387 123C0 D5               B=C    A              Save in B(A)
388 123C2 22               P=     2              # pointers to adjust
389 123C4 1B3A             D0=(5) =GSBSTK        Top of GOSUB stack
          5F2
390 123CB 80F0             CPEX   0              Set C(0)=# ptrs to adjust
391 123CF 21               P=     1              Set NO LEEWAY check
392 123D1 8F00             GOSBVL =PSHSTL        Push Rtn Lvl 1 with NO Leeway
          000
393 123D8 07               C=RSTK                Restore Return address
394 123DA ACB              C=D    S              Restore Return type
395 123DD 812              CSLC                  Move Return type to C(0)
396 123E0 15D5             DAT1=C 6              Write Return type& addr to stack
397 123E4 7180             GOSUB  Rstk<r         Restore 2 stack levels
398                *
399                * Initialize Relative Position in LEX Buffer
400                *
401 123E8 D1               B=0    A              Rel. Position in LEX Buffer
402                *
403                * Get LEX File POLL address
404                *
405 123EA 7480  POLL10 GOSUB  GLXPOL              Get LEX File POLL address
406 123EE 4C4              GOC    POLL40         No more LEX Files
407                *
408                * Save Relative Position in LEX Buffer
409                *
410 123F1 7301             GOSUB  FIRSAV         Position to First item in Save Area
411 123F5 145              DAT1=C A              Relative Positon in LEX Buffer
412                *
413                * Push LEX File Jump (Poll) address on stack
414                * Restore Poll3, A,D,D0,D1
415                *
416                * NOTE:  Making RESSVA a subroutine adds +12 cycles to this loop
417                *
418                *
419 123F8 D9               C=B    A              LEX File jump address
420 123FA 06               RSTK=C                Save on stack
421 123FC 7650             GOSUB  RESSVA         Restore Poll#,A,D,D0,D1
422 12400 07               C=RSTK                LEX File jump address
423                *
424                * Carry Clear from SNAPR*
425                * Clear XM flag
426                * Gosub to ROM Poll Routine
427                *
428 12402 821              XM=0                  Clear XM flag
429 12405 8F00             GOSBVL =CRGJMP        Gosub to Poll Routine
          000
430                *
431                * Clear B(S) to save carry
```

```
                        * If carry set
                        *    Error Response from ROM
                        *    Set A=C to preserve C
                        *
1240C AC1                       B=0    S              Carry indicator
1240F 461                       GOC    POLL20         Error response
                        *
                        * Increment B(S) to indicate Carry Clear return
                        * ROM responded "normally"
                        *
12412 B45                       B=B+1  S              Indicate Carry Clear
12415 831                       ?XM=0                 Normal response from LEX file ?
12418 11                        GOYES  POLL30
                        *
                        * Continue Poll
                        *    Restore Relative Position in LEX Buffer
                        *    go Read LEX Buffer
                        *
1241A 7AD0                      GOSUB  FIRSAV         Position to first item in Save Area
1241E 143                       A=DAT1 A              Relative Position in LEX Buffer
12421 D8                        B=A    A
12423 56C                       GONC   POLL10         go Read LEX Buffer for next LEX Fil
                        *
                        * Response from LEX File
                        *   Save C in A if Error response from Responder
                        *   Save current A,D,D0,D1 in SNAPBF
                        *   Restore Rtn Lvl 3, Rtn Lvl 2 and write Rtn Lvl 1 to SNAPBF
                        *   Release POLL save area
                        *   Restore current A,D,D0,D1
                        *   Move Carry setting to B(S)
                        *   Set C=A
                        *   Return indicating carry from LEX File
                        *
12426 AFA   POLL20  A=C    W              Save C in A if Error response
12429 7D70  POLL30  GOSUB  RESRTN         Restore Rtn levels, Release save ar
1242D 7E10          GOSUB  Snaprs         Restore current A,D,D0,D1
12431 06            RSTK=C                Rtn address of caller (Rtn Lvl 1)
12433 AF6           C=A    W              Set C=A on return
12436 A4D           B=B-1  S              Saved carry
12439 01            RTN                   Return indicating carry
                        *
                        * NO more LEX Files
                        *    Restore A,D,D0,D1 from SAVE stack
                        *    Save A,D,D0,D1,Rtn1 in SNAPBF, Restore Rtn2,3, Release Save
                        *    Restore A,D,D0,D1,Rtn1 from SNAPBF
                        *    Return to Calling Routine with XM=1
                        *
1243B 79B0  POLL40  GOSUB  FIRSAV         Position to first item in Save
1243F 7310          GOSUB  RESSVA         Restore A,D,D0,D1 on entry
12443 7360          GOSUB  RESRTN         Restore Return Address, Release sav
12447 7400          GOSUB  Snaprs         Restore A,D,D0,D1,Rtn1 from SNAPBF
1244B 06            RSTK=C                Rtn address of caller
1244D 00    =RTNSXM RTNSXM                Set XM on Return
1244F 8D00  =Snaprs GOVLNG =SNAPRS        Restore from SNAPBF
      000
```

```
                    *
                    * Restore A,D,D0,D1,Process# from SAVSTK - 20
                    *   D1 @ SAVSTK - 5
                    *
 12456 1CE   RESSVA D1=D1- (lRTN3p)+(lRTN2p)+(lPOL#p)
 12459 8D00 =snapr* GOVLNG =SNAPR*        Restore
       000
                    *
                    * Save and Restore 2 stack levels in RSTKBF
                    *
 12460 21    R<rstk P=     1              Save 2 Stack levels
 12462 8D00 =R<Rstk GOVLNG =R<RSTK
       000
 12469 21    Rstk<r P=     1              Restore 2 Stack levels
 1246B 8D00 =Rstk<R GOVLNG =RSTK<R
       000
```

```
                        STITLE Get LEX File Poll Address
                    ********************************************************************
                    ********************************************************************
                    **
                    ** Name:    GLXPOL  -  Get LEX File Poll Address
                    **
                    ** Category:   LOCAL
                    **
                    ** Purpose:
                    **      Get LEX File Poll Address
                    **
                    ** Entry:
                    **      B(A) = Relative Offset into LEX Buffer
                    **
                    ** Exit:
                    **      Carry Clear
                    **        C(A) = Updated Relative Offset into LEX Buffer
                    **        B(A) = LEX File POLL address
                    **
                    **      Carry set
                    **        No more LEX Files in Buffer
                    **        (LEX ID = 0 --> @ Mainframe LEX file)
                    **
                    ** Calls:      LXFND
                    **
                    ** Uses.......
                    **  Exclusive: A(A),B(A),C(A),P=0,D0,D1
                    **  Inclusive: A(A),B(A),C(A),P=0,D0,D1
                    **
                    ** Stl lvls:   1
                    **
                    ** Detail:
                    **      Find LEX Buffer              (LXFND)
                    **      Add relative offset to buffer start
                    **      Read LEX ID
                    **      If LEX ID=0
                    **         RTNYES
                    **      else
                    **         Read LEX File's Main Table Address
                    **         Read Relative POLL address
                    **         Compute Absolute POLL address
                    **         Update Relative position in Buffer
                    **         RTNCC
                    **
                    ** History:
                    **
                    **    Date     Programmer             Modification
                    **   --------   ----------   ---------------------------------
                    **   07/13/82   JP           Modified documentation
                    **   09/18/82   JP           Removed "=" sign from name
                    **
                    ********************************************************************
                    ********************************************************************
12472 8F00  GLXPOL GOSBVL =LXFND         Find table
      000
```

```
553 12479 137          CD1EX                 C <-- LEX Buffer start
554 1247C C9           C=C+B  A              Current Position in Buffer
555 1247E 135          D1=C
556              *
557              * Read LEX ID
558              *   If Main Table ID  (00)
559              *      Return Carry
560              *
561 12481 14F          C=DAT1 B              Read LEX ID
562 12484 96A          ?C=0   B              Main Table ID ?
563 12487 00           RTNYES
564              *
565              * Skip LEX ID and Token range
566              * Read LEX File's MAIN Table Address
567              * Read Relative POLL Address
568              * Compute Absolute POLL Address
569              *
570 12489 175          D1=D1+ (1LXID)+(1LXTKR) Skip LEX ID and Token Range
571 1248C 147          C=DAT1 A              LEX Id's MAIN Table
572 1248F 134          DO=C
573 12492 184          DO=DO- 5              Ptr to Relative POLL Addr
574 12495 146          C=DATO A              Relative POLL Address
575 12498 132          ADOEX                 Position in LEX File
576 1249B C2           C=C+A  A              Absolute POLL Address
577              *
578              * Update and Save Relative Position in LEX Buffer
579              *
580 1249D DD           CBEX   A              B <- POLL Addr, C <- Offset
581 1249F 136          CDOEX
582 124A2 16A          DO=DO+ 1LXENT         Offset = Offset + Length of entry
583 124A5 136          CDOEX
584 124A8 03           RTNCC
```

```
585                 STITLE Restore Sub Levels, Adjust AVMEME
586               ********************************************************************
587               ********************************************************************
588               **
589               ** Name:    RESRTN  -  Restore Sub Lvls 2,3, release save area
590               **
591               ** Category:   LOCAL
592               **
593               ** Purpose:
594               **      Save Current A,D,D0,D1 in SNAPBF
595               **      Restore subroutine levels, preserving calling return addr
596               **      Write Rtn Lvl 1 to SNAPBF
597               **      Release Poll Save area
598               **
599               ** Entry:
600               **
601               ** Exit:
602               **      Through SRLEAS
603               **      2 subroutine levels restored
604               **      Rtn level 1 popped from GOSUB stack and written to SAVSTK
605               **      Save area released, AVMEME updated
606               **      Returns CC
607               **
608               ** Calls:      FIRSAV,SRLEAS,SNAPSV,R<RSTK,RSTK<R ,POPUPD
609               **
610               ** Uses.......
611               **      A,B(A),C(A),D0,D1
612               **
613               ** Stk lvls:   0
614               **
615               ** Detail:
616               **
617               **      SRLEAS uses A(A),B(A),C(A),D0,D1
618               **
619               **      POLL Saved Information:
620               **                                                    Low Memory
621               ** lAp       A                                  16 @ SAVSTK-62
622               ** lDp       D                                  16      |
623               ** lD1p      D1                                  5      |
624               ** lD0p      D0                                  5      |
625               ** lPOL#p    Poll#                               5      v
626               ** lRTN2p    Return Level 2                      5
627               ** lRTN3p    Return Level 3                      5
628               ** lBPOSp    Relative Positon in LEX Buffer      5 @ SAVSTK-5
629               **                                                    High Memory
630               **      GOSUB Stack:
631               **              ---------------
632               ** GSBSTK --> |F|Rtn Addr 1 |                      6
633               **              ---------------
634               **
635               ** Algorithm:
636               **
637               **      Pop calling return address off stack
638               **      Save A,D,D0,D1,Calling Rtn address in SNAPBF (SNAPSV)
639               **      Restore Rtn Level 2,3 from SAVSTK
```

```
640                **      Save 2 subroutine levels in RSTKBF          (R<RSTK)
641                **      Pop Rtn Lvl 1 from GOUSB stack              (POPUPD)
642                **      Restore 2 subroutine levels from RSTKBF     (RSTK<R)
643                **      Read Calling Return address from SNAPBF+42
644                **      Write Level 1 Return Address to SNAPBF
645                **      Restore calling return address to stack
646                **      Release Save Area, adjust AVMEME            (SRLEAS)
647                **
648                ** History:
649                **
650                **    Date     Programmer            Modification
651                **   --------  ----------  ----------------------------------
652                **   10/14/82  JP          Rewrote to interface to SNAPBF
653                **   02/05/83  JP          Rewrote to interface to GOSUB stack
654                **   09/18/83  JP          Removed "=" from routine name
655                **
656                ********************************************************************
657                ********************************************************************
658 124AA 07     RESRTN C=RSTK               Pop calling address off stack
659 124AC 8F00          GOSBVL =SNAPSV        Save A,D,D1,DO in SNAPBF
           000
660                *
661                * Restore Return Levels 2,3 from SAVSTK
662                *
663 124B3 7140          GOSUB  FIRSAV         Pos to 1st item of saved info
664 124B7 1C4           D1=D1- (1RTN3p)       Move to Rtn Level 3
665 124BA 147           C=DAT1 A
666 124BD 06            RSTK=C                Restore Rtn Level 3
667 124BF 1C4           D1=D1- (1RTN2p)       Move to Rtn Level 2
668 124C2 147           C=DAT1 A
669 124C5 06            RSTK=C                Restore Rtn Level 2
670                *
671                * Save 2 stack levels before POPUPD call
672                * Pop Rtn Level 1 from GOSUB stack
673                * Restore 2 stack levels
674                *
675 124C7 759F          GOSUB  R<rstk         Save 2 stack levels
676 124CB 8F00          GOSBVL =POPUPD        Pop GOSUB stack
           000
677 124D2 739F          GOSUB  Rstk<r         Restore 2 stack levels
678                *
679                * Read calling return address from this RESRTN call
680                *   Set D1 @ Return address field (SNAPBF + 42 )
681                * Move Return Addr 1 to C(A)
682                * Write Rtn Addr 1 to SNAPBF area
683                * Push calling return address on stack
684                *
685 124D6 1FA1          D1=(5) (=SNAPBF)+42
           8F2
686 124DD 143           A=DAT1 A              Calling routine address
687 124E0 DB            C=D    A              Return Addr 1
688 124E2 DE            ACEX   A              A <-- Rtn Addr 1; C <-- Calling Rtn
689 124E4 141           DAT1=A A              Write Rtn addr 1 to SNAPBF
690 124E7 06            RSTK=C                Push Calling routine address on sta
691                *
```

```
692             * Release POLL Save Area, Update AVMEME
693             *
694 124E9 20            P=      0
695 124EB D2            C=0     A
696 124ED 31E3          LC(2)   1POLSV          Length of POLL Save Area
697 124F1 8D00          GOVLNG  =SRLEAS
          000
```

```
                        STITLE BEGSAV - Beginning of Saved Information
                ********************************************************************
                ********************************************************************
                **
                ** Name:    FIRSAV  -  Position to First Item in Save Area
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Position to First Item of Saved POLL Information
                **
                ** Entry:
                **
                ** Exit:
                **      D1 @ Start of Save info - 1BPOSp
                **         @ Start of Relative offset into LEX Buffer Entry
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Exclusive: A,D1
                **  Inclusive: A,D1
                **
                ** Stk lvls:   0
                **
                ** Detail:
                **      Saved information for POLL:                    Low Memory
                **
                ** 1A        A                                    16 @ SAVSTK-62
                ** 1D        D                                    16
                ** 1D1p      D1                                    5     |
                ** 1D0p      D0                                    5     |
                ** 1POL#p    Poll#                                 5     v
                ** 1RTN2p    Return Level 2                        5
                ** 1RTN3p    Return Level 3                        5
                ** 1BPOSp    Relative Positon in LEX Buffer        5 @ SAVSTK
                **                                                     High Memory
                **
                ** History:
                **
                **    Date      Programmer             Modification
                **   --------   ----------   ----------------------------------
                **   07/13/82   JP           Modified documentation
                **   10/14/82   JP           Interface to SNAPBF, new save order
                **   02/05/83   JP           Modified doc for new save infor
                **   09/18/83   JP           Removed "=" from routine name
                **
                ********************************************************************
                ********************************************************************
124F8 1FE9 5F2  FIRSAV D1=(5) =SAVSTK
124FF 143               A=DAT1 A
12502 131               D1=A                  Start of saved information
12505 1C4               D1=D1- 1BPOSp         Position to start of first entry
12508 01                RTN
```

```
752                 STITLE Fast POLL
753           ********************************************************************
754           ********************************************************************
755           **
756           ** Name:(S) FPOLL   -  Fast Poll all LEX files with Process #
757           **
758           ** Category:   POLL
759           **
760           ** Purpose:
761           **      Poll LEX Files FAST, nothing is saved
762           **
763           ** Entry:
764           **      Process# @ Calling Routine Return Address
765           **      Process# = CON(2)
766           **
767           **      Example:
768           **
769           **      GOSBVL =FPOLL
770           **      CON(2) =pMNLP     Main Loop Fast Poll
771           **
772           **      At entry to LEX File POLL routine:
773           **         Carry Set to indicate FAST Poll
774           **         B(A) = Process#
775           **                B(B)   = Process#
776           **                B(2-4) = 0
777           **         D(A) = Relative Position in LEX Buffer
778           **                Must be preserved ALWAYS !!!!!
779           **                  If the Poll Handler is responding
780           **                  and handling the poll such that
781           **                  the Poll will stop: D may be used.
782           **         R0,R1,R2,R3 intact
783           **         A LEX File may not destroy R0-R3 while determining whe
784           **           to respond.  Individual POLL routines must be checke
785           **           register usage when responding.
786           **
787           **         Stack levels are 2 deeper than caller
788           **
789           ** Exit:
790           **      P=0
791           **         Assuming no LEX File has set P
792           **      XM=0
793           **         Process has been handled by LEX File
794           **
795           **      XM=1
796           **         No response to Poll
797           **
798           **      If a LEX File wants the Poll to continue to others
799           **         XM=1 on return
800           **         Registers needed to be passed to other LEX Files
801           **         must be preserved !!!!!
802           **
803           **      Typically a fast poll must continue to ALL LEX files
804           **
805           **
806           ** Calls:      GLXPOL
```

```
807                ** 
808                ** Uses.......
809                **  Exclusive: A(A),B(A),C(A),D(A),D0,D1
810                **  Inclusive: A(A),B(A),C(A),D(A),D0,D1
811                ** 
812                **  D(A) cannot be destroyed by any LEX File
813                **  R0-R3, status must remain intact while determining if
814                **    reponding to poll.
815                ** 
816                ** Stk lvls:   2
817                ** 
818                ** Algorithm:
819                ** 
820                **      Initialize Relative Offset to LEX Buffer to 0
821                ** 1:   Get LEX File Poll Address              (GLXPOL)
822                **      If LEX File Poll Address         (Carry clear)
823                **         Save Relative position in LEX Buffer   (D)
824                **         Retrieve Process #
825                **         Clear XM
826                **         Gosub to LEX File's Poll routine w/ Carry set
827                **         If LEX file did not respond          (XM=1)
828                **            Restore relative position in LEX buffer
829                **            goto 1;
830                **         else
831                **            Adjust Return Address past Process#
832                **            RTN
833                **      else
834                **         Adjust Return Address past Process#
835                **         RTNSXM
836                ** 
837                ** History:
838                ** 
839                **    Date     Programmer              Modification
840                **  --------   ----------   ---------------------------------
841                **  07/13/82   JP           Modified documentation
842                **  06/09/82   JP           Packed out CRGJMP/set carry:FPOL40
843                ** 
844                ********************************************************************
845                ********************************************************************
846 1250A D1    =FPOLL  B=0    A               Rel. Position in LEX Buffer
847 1250C 726F  FPOL10 GOSUB  GLXPOL           Get LEX Poll Address
848 12510 452          GOC    FPOL20          No more LEX Files
849 12513 D7           D=C    A               Save Rel. Position in Buffer
850                *
851                * Retrieve Process #
852                *
853 12515 07           C=RSTK                 Calling routine address
854 12517 06           RSTK=C
855 12519 135          D1=C                   Process# @ Return Address
856 1251C D2           C=0    A
857 1251E 14F          C=DAT1 B               Read Process#
858 12521 DD           BCEX   A               B <-- Process#, C <-- Poll Address
859                *
860                * Clear XM flag
861                * Gosub to LEX File Poll handler
```

```
                *   Set carry in the process
                *
12523 821               XM=0
12526 7C10              GOSUB  FPOL40           Gosub to Poll handler w/ Carry set
                *
                * If LEX File responded
                *    Adjust Return address past Process# & Return
                *
1252A 831               ?XM=0                   Response ?
1252D 31                GOYES  FPOL30
                *
                * Retrieve Relative Position in LEX Buffer
                * go POLL next LEX File
                *
1252F DB                C=D    A                Relative Pos in LEX Buffer
12531 D5                B=C    A                Rel Position in LEX File
12533 58D               GONC   FPOL10           B.E.T.
                *
                * No more LEX Files
                *    Adjust Return address past Process #
                *    RTNSXM
                *
12536 07    FPOL20      C=RSTK                  Return Address @ Process#
12538 E6                C=C+1  A
1253A E6                C=C+1  A                Position past Process#
1253C 06                RSTK=C
1253E 00                RTNSXM
                *
                * LEX File Responed
                *    Adjust Return address past Process #
                *
12540 07    FPOL30      C=RSTK                  Return Address @ Process#
12542 E6                C=C+1  A
12544 E6                C=C+1  A
                *
                * Subroutine for jump to Poll handler
                *   Push Poll handler address on stack
                *   Set carry to indicate Fast Poll
                *
12546 06    FPOL40      RSTK=C
12548 02                RTNSC
```

```
 CRGJMP  Ext                    -   429
 FIRSAV  Abs    75000 #124F8    -   747   410   450   479   663
 FPOL10  Abs    75020 #1250C    -   847   878
 FPOL20  Abs    75062 #12536    -   884   848
 FPOL30  Abs    75072 #12540    -   893   871
 FPOL40  Abs    75078 #12546    -   901   865
=FPOLL   Abs    75018 #1250A    -   846
 GLXPOL  Abs    74866 #12472    -   552   405   847
 GSBSTK  Abs   193955 #2F5A3    -    13   389
 LXFND   Ext                    -   552
 MEMCKL  Ext                    -   311
 Moveu3  Ext                    -   345
 POLERR  Abs    74523 #1231B    -   282   312
=POLL    Abs    74551 #12337    -   300
 POLL10  Abs    74730 #123EA    -   405   453
 POLL20  Abs    74790 #12426    -   465   437
 POLL30  Abs    74793 #12429    -   466   444
 POLL40  Abs    74811 #1243B    -   479   406
=POLLD+  Abs    74541 #1232D    -   292
 POPUPD  Ext                    -   676
 PSHSTL  Ext                    -   392
 R<RSTK  Ext                    -   496
=R<Rstk  Abs    74850 #12462    -   496
 R<rstk  Abs    74848 #12460    -   495   376   675
 RESRTN  Abs    74922 #124AA    -   658   466   481
 RESSVA  Abs    74838 #12456    -   490   421   480
=RMEM    Abs    74533 #12325    -   286
 RSTK<R  Ext                    -   498
=RTNSXM  Abs    74829 #1244D    -   484
=Rstk<R  Abs    74859 #1246B    -   498
 Rstk<r  Abs    74857 #12469    -   497   397   677
 SALLOC  Ext                    -   329
 SAVSTK  Abs   193950 #2F59E    -    13   747
 SNAPBF  Abs   194544 #2F7F0    -    13   342   685
 SNAPR*  Ext                    -   491
 SNAPRS  Ext                    -   485
 SNAPSV  Ext                    -   301   659
 SRLEAS  Ext                    -   697
=Snaprs  Abs    74831 #1244F    -   485   282   467   482
 eMEM    Ext                    -   286
 lBPOSp  Abs        5 #00005    -    12   750
 lLXENT  Abs       11 #0000B    -    12   582
 lLXID   Abs        2 #00002    -    12   570
 lLXTKR  Abs        4 #00004    -    12   570
 lPOL#p  Abs        5 #00005    -    12   370   490
 lPOLSV  Abs       62 #0003E    -    12   294   309   327   696
 lPOLra  Abs        6 #00006    -    12   294
 lRTN1p  Abs        5 #00005    -    12   354
 lRTN2p  Abs        5 #00005    -    12   373   490   667
 lRTN3p  Abs        5 #00005    -    12   490   664
=snapr*  Abs    74841 #12459    -   491
```

Input Parameters

  Source file name is JP&POL::MS

  Listing file name is JP/POL:TI:ML::-1

  Object file name is JP%POL:TI:MS::-1

```
                                111111
                       0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    M   M  N   N   &     TTTTT  M   M
 2            *    MM MM  N   N  & &      T    MM MM
 3            *    M M M  NN  N  & &      T    M M M
 4            *    M M M  N N N   &       T    M M M
 5            *    M   M  N  NN  & & &    T    M   M
 6            *    M   M  N   N  &  &     T    M   M
 7            *    M   M  N   N   && &    T    M   M
 8            *
 9                      TITLE  Clock System <831228.1028>
10 1254A                ABS    #1254A
11                      RDSYMB SB%RAM::MS
```

```
12                      STITLE Low-level routines
13                ****************************************************************
14                ** Start of clock system low-level timekeeping routines.
15                ****************************************************************
16                *
17                ****************************************************************
18                ** The following storage is used in the internal clock
19                ** system:
20                **
21                **      name      size(nibs)     function
22                **      ----      ----------     --------
23                **      NXTIRQ       12          time of next sreq
24                **      ALRM1        12          on timer #1
25                **      ALRM2        12          on timer #2
26                **      ALRM3        12          on timer #3
27                **      ALRM4        12          timeout
28                **      ALRM5        12          pause
29                **      ALRM6        12          external alarm
30                **      PNDALM        2          bitmap of pending alarms
31                **      TIMOFS       12          time error offset for AF use
32                **      TIMLST       12          time of last exact
33                **      TIMLAF       12          time of last AF correction
34                **      TIMAF         6          accuracy factor
35                **
36                ** All times are stored in units of 1/512th seconds.
37                **
38                ** ----------
39                **
40                ** Software timebase correction is used to compensate for
41                ** the estimated .005% inaccuracy of the crystal timebase.
42                ** An accuracy factor (AF) specifies the accuracy of the
43                ** clock, and is defined as:
44                **
45                **          actual time = timer time(1+1/AF)
46                **
47                ** In other words, the accuracy factor tells how many
48                ** seconds should elapse before adding a second to clock
49                ** time.  A positive accuracy factor will correct a slow
50                ** clock, a negative AF will correct a fast clock.
51                **
52                ** ----------
53                **
54                ** The external alarm is for use by external modules
55                ** (such as pocket secretary or a controller) which may
56                ** wish to schedule events.  The exact format of how each
57                ** external module maintains its alarm stack is
58                ** unimportant, as long as the following protocol is
59                ** observed:
60                **
61                **   The module may schedule an alarm if either:
62                **      1) Current external alarm is past due.
63                **      2) Current external alarm occurs after alarm
64                **         the code wishes to schedule.
65                **
66                ** When an external alarm comes due, the system will
```

```
67          ** poll and reset the external alarm pending bit.  Each
68          ** module capable of scheduling alarms should then
69          ** compare the current time to its alarm stack to
70          ** determine if any of its alarms are pending.
71          **
72          ************************************************************************
73          *
74          ************************************************************************
75          ************************************************************************
76          **
77          ** Name:    GETTIM  -  Read Time System's Countdown Timer
78          **
79          ** Category:   LOCAL
80          **
81          ** Purpose:
82          **      Read countdown timer #2.
83          **
84          ** Entry:
85          **      None.
86          **
87          ** Exit:
88          **      Timer value in A, with sign-extend to full word.
89          **      Carry set if negative, else carry clear.
90          **      P=5.
91          **
92          ** Calls:      None.
93          **
94          ** Uses.......
95          **             A,B,P
96          **
97          ** Stk lvls:   0
98          **
99          ** Detail:
100         **      512 hz countdown timer operates asynchrously of CPU
101         **      clock.  It can decrement at any time, including during
102         **      a read.  The double-read algorithm assures that we
103         **      do not encounter situations such as follows:
104         **        Timer = 120000, decrements after CPU reads first
105         **        three digits, yielding a reading of 11F000 (since
106         **        CPU reads LSN to MSN).
107         **
108         ** Algorithm:
109         **      Read timer -> B.
110         **      Read timer -> A.
111         **      If A#B then Read timer -> A.
112         **      B=A.
113         **      If bit 23 of B is set, then perform sign extend on A
114         **        from 6-nibble quantity to full-word and RTNSC, else
115         **        RTNCC.
116         **
117         ** History:
118         **
119         **     Date     Programmer            Modification
120         **   --------   ----------   ---------------------------------
121         **   06/03/82   NM           Added documentation
```

```
122                **
123                ********************************************************************
124                ********************************************************************
125 1254A 04     GETTIM SETHEX
126 1254C 1B8F         DO=(5) =TIMER2        Point to timer in disp driv.
          2E2
127 12553 AF0          A=0    W
128 12556 15A5         A=DATO 6              Read timer
129 1255A A98          B=A    WP
130 1255D 15A5         A=DATO 6              Read again.
131 12561 25           P=     5
132 12563 910          ?A=B   WP             Kerchunk anywhere?
133 12566 60           GOYES  GETTM2         No.  Reads are good.
134 12568 15A5         A=DATO 6              Yes.  One more read.
135 1256C A88    GETTM2 B=A    P
136 1256F A05          B=B+B  P              Negative?
137 12572 500          RTNNC                 No
138 12575 B98          A=-A   WP             Yes.  perform sign-extend
139 12578 BF8          A=-A   W
140 1257B 02           RTNSC
141                ********************************************************************
142                ********************************************************************
143                **
144                ** Name:    ALMSRV  -  Check For Pending Alarms
145                **
146                ** Category:   TIME
147                **
148                ** Purpose:
149                **      Service clock system when alarm becomes pending.
150                **      Build and return bitmap of pending alarms.
151                **
152                ** Entry:
153                **      None.
154                **
155                ** Exit:
156                **      Bitmap of pending alarms in status bits.
157                **        S0 = alarm 1
158                **        S1 = alarm 2
159                **             . . .
160                **        S5 = alarm 6
161                **      If S0, S1, S2 or S5 is set, S12 is also set to
162                **        indicate an exception condition.
163                **      Carry clear if clock was not timed out.
164                **      Carry set if clock was timed out.
165                **      If carry set, C[W]=current time in 512ths.
166                **      P=0.
167                **
168                ** Calls:      GETPND, GETTIM, ST01, CMPT, RC01.
169                **
170                ** Uses.......
171                **             A,B,C,D,D0,P,D1,S0-S11, SCRTCH[0-31] (only
172                **               if carry set (clock was timed out))
173                **
174                ** Stk lvls:   2
175                **
```

```
176                ** Algorithm:
177                **      Read PNDALM; Set S12 if bits 0,1,2 or 5 set.
178                **      Read timer.  Return if not negative.
179                **      Stash R0-R3 in SCRTCH.
180                **      Call CMPT (performs update).
181                **      Restore R0-R3.
182                **
183                ** History:
184                **
185                **     Date     Programmer            Modification
186                **   --------   ----------   ---------------------------------
187                **   06/03/82   NM           Added documentation
188                **
189                ********************************************************************
190                ********************************************************************
191 1257D 0B    =ALMSRV CSTEX
192 1257F 7AB4          GOSUB  GETPND
193 12583 7510          GOSUB  SF12?
194 12587 7FBF          GOSUB  GETTIM          Timer negative?
195 1258B 20            P=     0
196 1258D 500           RTNNC                 Return if not
197 12590 7815          GOSUB  ST01           Stash R0 & R1.
198 12594 7A10          GOSUB  CMPT           Perform update
199 12598 6D25          GOTO   RC01           Retrieve R0 & R1.
200                ********************************************************************
201                ********************************************************************
202                **
203                ** Name:    SF12?   -  Conditionally Set Exception Flag
204                **
205                ** Category:   LOCAL
206                **
207                ** Purpose:
208                **      Set S12 if a non-PAUSE, non-TIMEOUT alarm is pending.
209                **
210                ** Entry:
211                **      C[B] contains PNDALM bitmap.
212                **
213                ** Exit:
214                **      Flag 12 set if bits 0-2 or bit 5 of C[B] is set.
215                **      P=0.
216                **      ST[11-0] = C[X] on entry.
217                **
218                ** Calls:      None.
219                **
220                ** Uses.......
221                **             ST[0-11],P,C[0].
222                **
223                ** Stk lvls:   0
224                **
225                ** History:
226                **
227                **    Date     Programmer            Modification
228                **  --------   ----------   ---------------------------------
229                **  10/01/82   NM           Stripped out of other code
230                **
```

```
                ****************************************************************
                ****************************************************************
1259C 0A    SF12?  ST=C                  Alarm flags to ST
1259E 20           P=     0
125A0 A06          C=C+C  P              Shift off bit 3
125A3 90E          ?C#0   P              Alarms 0-2 pending?
125A6 70           GOYES  SF12?1         Yes
125A8 865          ?ST=0  5              No.  Alarm 5 pending?
125AB 00           RTNYES                No.
125AD 85C   SF12?1 ST=1   12             Set something funny flag.
125B0 01           RTN
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) CMPT    -  Return Current Time
                **
                ** Category:   TIME
                **
                ** Purpose:
                **      Read current time in 512ths since time 0.
                **
                ** Entry:
                **      None.
                **
                ** Exit:
                **      Current time in C and R1 (HEX ticks).
                **        (Time represented as # of 512ths sec since midnight
                **         1 Jan 0000).
                **      R0 = TIMER value corresponding to current time.
                **      HEX mode.
                **      Carry clear.
                **      P=0.
                **
                ** Calls:      GETTIM, GETIRQ, GETLAF, GETAF, IDIV, PUTLAF,
                **             CLKUPD (falls through).
                **
                ** Uses.......
                **             A,B,C,D,P,R0,R1,D0,D1,S0-S11
                **
                ** Stk lvls:   1
                **
                ** Detail:
                **      Routine computes current time (NXTIRQ-TIMER) and places
                **      value of TIMER corresponding to current time in R0.
                **      Then accuracy factor corrections are computed and the
                **      code falls through to CLKUPD to perform an update.
                **
                ** Algorithm:
                **      Read TIMER; save in R0.
                **      Read NXTIRQ; current time = NXTIRQ-TIMER; save in R1.
                **      Read TIMLAF; compute #ticks since last AF correction
                **        (TIME-TIMLAF); stash in D.
                **      Compute (TIME-TIMLAF)/abs(AF); quotient to A;
                **        remainder to B.
                **      Compute #ticks from old TIMLAF to new TIMLAF =
```

```
286             **        (TIME-TIMLAF)-REMAINDER -> D.
287             **      Negate A (quotient from division) if AF is negative.
288             **      [At this point, A=time correction, D=#ticks from old
289             **        TIMLAF to new TIMLAF.]
290             **      TIMLAF = TIMLAF + A + D.
291             **      TIME = TIME (from R1) + A.  Store TIME in R1.
292             **      Fall through to CLKUPD.
293             **
294             ** History:
295             **
296             **     Date     Programmer            Modification
297             **   --------   ----------   ---------------------------------
298             **   06/07/82   NM           Added documentation
299             **
300             ****************************************************************
301             ****************************************************************
302 125B2 749F =CMPT   GOSUB  GETTIM         Timer (in 1/512 secs) to a
303 125B6 100        R0=A                   Copy to r0.
304 125B9 77F3       GOSUB  GETIRQ          Next interrupt time to c
305 125BD B72        C=C-A  W               Current time in 1/512 secs
306 125C0 109        R1=C
307             * CODE FOR ACCURACY FACTOR CORRECTION STARTS HERE
308 125C3 AFA        A=C    W
309 125C6 7634       GOSUB  GETLAF          Time of last AF correction to c
310 125CA B7A        A=A-C  W               Ticks since last AF correction
311 125CD AF6        C=A    W
312 125D0 AF7        D=C    W               Save
313 125D3 73F3       GOSUB  GETAF           AF to c
314 125D7 7C32       GOSUB  SOCNEG
315 125DB 97E        ?C#0   W               AF exist?
316 125DE 50         GOYES  CMPT20
317 125E0 A7E        C=C-1  W               No, set to infinity (f's)
318 125E3 73F6  CMPT20 GOSUB  idiv          (ticks since timlaf)/AF
319 125E7 B73        D=D-C  W               Ticks from old to new timlaf
320 125EA 860        ?ST=0  0               AF positive?
321 125ED 50         GOYES  CMPT30          Yes
322 125EF BF8        A=-A   W               No, negate quotient
323 125F2 7A04  CMPT30 GOSUB  GETLAF
324 125F6 A7B        C=C+D  W               Ticks to new timlaf
325 125F9 A72        C=C+A  W               AF correction
326 125FC 7A14       GOSUB  PUTLAF
327 12600 119        C=R1                   Current time
328 12603 A72        C=C+A  W               AF correction
329 12606 109        R1=C
330 12609 66B0       GOTO   CLKUPD          Store new time
331             * END OF ACCURACY FACTOR CORRECTION CODE
332             ****************************************************************
333             ****************************************************************
334             **
335             ** Name:    CMPTIM  -  Compute Time In Secs Since Time 0
336             ** Name:    TIMRND  -  Truncate Time To Seconds
337             **
338             ** Category:   TIME
339             **
340             ** Purpose:
```

```
                **      Compute current time in seconds.
                **
                ** Entry:
                **   CMPTIM: None.
                **   TIMRND: C = time in ticks (512ths sec).
                **
                ** Exit:
                **      C = time in secs (HEX).
                **      R1 = Time in 512ths (HEX) (CMPTIM ONLY).
                **      R0 = Timer value corresponding to time (CMPTIM ONLY).
                **      HEX mode.
                **      Carry clear.
                **
                ** Calls:      CMPT.
                **
                ** Uses.......
                **              A,B,C,D,P,R0,R1,D0,D1,S0-S11
                **
                ** Stk lvls:   2
                **
                ** Algorithm:
                **      CMPT.
                **      SR 9 bits.
                **
                ** History:
                **
                **     Date     Programmer            Modification
                **   --------   ----------   --------------------------------
                **   06/07/82   NM           Added documentation
                **
                ********************************************************************
                ********************************************************************
1260D 71AF      =CMPTIM GOSUB  CMPT           Time in 1/512 sec to c
12611 BF6       =TIMRND CSR    W
12614 BF6               CSR    W
12617 81E               CSRB                  Must sr 9 bits.
1261A 01                RTN
                ********************************************************************
                ********************************************************************
                **
                ** Name:    COMPAF  -  Compute Accuracy Factor
                **
                ** Category:   TIME
                **
                ** Purpose:
                **      Compute a new accuracy factor for EXACT function.
                **
                ** Entry:
                **      R1 = Current time.
                **      (Other inputs are in clock system registers).
                **      HEX mode.
                **
                ** Exit:
                **      Carry clear: Illegal AF (< 10).
                **      Carry set:
```

```
396             **       A = AF (HEX), with sign extended to full word.
397             **
398             ** Calls:      GETLST, GETOFS, GETAF, MPY, IDIV, CHKAF (falls
399             **             through).
400             **
401             ** Uses.......
402             **             A,B,C,D,P,D1,S0
403             **
404             ** Stk lvls:   1
405             **
406             ** Detail:
407             **
408             **   AF is number of seconds to pass before adding or
409             **   subtracting a second.
410             **
411             **   Define: tr=real elapsed time,
412             **           ti=internal elapsed time ('ticks'),
413             **
414             **   Then the definition of accuracy factor is:
415             **                  tr=ti(1+1/AF).
416             **
417             **   The accuracy factor is thus computed as:
418             **                  AF:=ti/(tr-ti).
419             **
420             **   A value of AF=0 is used to represent the case of
421             **   AF=infinity.
422             **
423             ** Algorithm:
424             **
425             **   Since
426             **         tr=currtime - timlst
427             **   and
428             **         ti=(currtime-timlst-timofs)/(1+1/AF),
429             **
430             **   the expression for the new accuracy factor is:
431             **
432             **               AF*(currtime-timlst-timofs)
433             **        AF' = ---------------------------
434             **               currtime-timlst+AF*timofs
435             **
436             **
437             **             = (currtime-timlst-timofs)/timofs
438             **
439             **               if AF=0.
440             **
441             ** History:
442             **
443             **     Date     Programmer            Modification
444             **   --------   ----------   ---------------------------------
445             **   06/07/82   NM           Added documentation
446             **
447             ********************************************************************
448             ********************************************************************
449 1261C 111   COMPAF A=R1                  Current time
450 1261F 7404        GOSUB  GETLST
```

```
451 12623 B7A          A=A-C  W              Currtime-timlst
452 12626 7063         GOSUB  GETOFS
453 1262A B7A          A=A-C  W              Currtime-timlst-timofs
454 1262D AF7          D=C    W              Hold timofs
455 12630 7693         GOSUB  GETAF
456 12634 97E          ?C#0   W              AF=0?
457 12637 80           GOYES  COMP10         No
458 12639 AFB          C=D    W              Yes, new AF=a/c
459 1263C 572          GONC   COMP20         B.E.T.
460 1263F 7D96 COMP10  GOSUB  mpy            AF(currtime-timlst-timofs)
461 12643 AFF          CDEX   W              Stash.  retrieve timofs
462 12646 AFA          A=C    W
463 12649 7D73         GOSUB  GETAF
464 1264D 7F86         GOSUB  mpy            AF*timofs
465 12651 AFF          CDEX   W
466 12654 AFA          A=C    W              Position numerator
467 12657 7CC3         GOSUB  GETLST
468 1265B B73          D=D-C  W              AF*timofs-timlst
469 1265E 119          C=R1
470 12661 A7B          C=C+D  W              Currtime-timlst+AF*timofs
471 12664 97A  COMP20  ?C=0   W              Denominator=0?
472 12667 93           GOYES  CHKA25         Yes.  result=0 (infinity)
473 12669 840          ST=0   0              Mark result positive
474 1266C 942          ?A=C   S              Numerator & denom same signs?
475 1266F 80           GOYES  COMP40         Yes
476 12671 850          ST=1   0              No.  result negative.
477 12674 BFA          C=-C   W              Make terms same sign
478 12677 948  COMP40  ?A=0   S              Terms positive?
479 1267A 80           GOYES  COMP50         Yes
480 1267C BF8          A=-A   W              No.  negate both terms.
481 1267F BFA          C=-C   W
482 12682 7456 COMP50  GOSUB  idiv           New AF.  fall through to chkaf
483 12686 978          ?A=0   W              Computed AF way too much?
484 12689 22           GOYES  CHKA40         Yes.  Will RTNCC.
485             ****************************************************************
486             ****************************************************************
487             **
488             ** Name:   CHKAF   -  Check For Valid Accuracy Factor.
489             **
490             ** Category:   TIME
491             **
492             ** Purpose:
493             **      Check that accuracy factor lies in the range:
494             **              800000H <= AF <= 7FFFFFH.
495             **
496             ** Entry:
497             **      A = ABS(AF).
498             **      S0=0 if AF positive, 1 if AF negative.
499             **
500             ** Exit:
501             **      Carry clear: ABS(AF)<10 and AF<>0 (invalid value).
502             **      Carry set:
503             **        A=AF in 2's complement (sign extended to whole
504             **          word).
505             **        If AF>7FFFFF or AF<800000, returns AF=0.
```

```
506                 **
507                 ** Calls:      None.
508                 **
509                 ** Uses.......
510                 **  Exclusive: A,C,P
511                 **
512                 ** Stk lvls:   0
513                 **
514                 ** Algorithm:
515                 **      C=(if S0=0 ?FFFFF else 800000) {max abs AF}.
516                 **      AF < max?  Set AF=0 if not {represents AF=infinity}.
517                 **      ABS(AF) < 10?  RTNCC if yes.
518                 **      If S0=1, negate AF.
519                 **      RTNSC.
520                 **
521                 ** History:
522                 **
523                 **     Date     Programmer            Modification
524                 **   --------   ----------   ---------------------------------
525                 **   06/07/82   NM           Added documentation
526                 **
527                 ********************************************************************
528                 ********************************************************************
529                 **
530                 ** NOTE: Above code falls through
531                 **
532 1268B AF2   CHKAF  C=0    W
533 1268E 25           P=     5
534 12690 308          LCHEX  8              Max abs(AF) if neg
535 12693 870          ?ST=1  0              AF neg?
536 12696 50           GOYES  CHKA20         Yes
537 12698 A1E          C=C-1  WP             No.  max AF.
538 1269B 9FA   CHKA20 ?A<=C  W              AF<=max?
539 1269E 50           GOYES  CHKA30         Yes
540 126A0 AF0   CHKA25 A=0    W              No.  use zero.
541 126A3 978   CHKA30 ?A=0   W
542 126A6 00           RTNYES
543 126A8 A92          C=0    WP
544 126AB 20    CHKA40 P=     0              Entry to RTNCC from above
545 126AD 309          LCHEX  9              Minimum AF-1
546 126B0 B72          C=C-A  W
547 126B3 500          RTNNC                 Return if < minimum
548 126B6 860          ?ST=0  0              Positive
549 126B9 00           RTNYES                Yes.  done.
550 126BB BF8          A=-A   W              No.  negate.
551 126BE 02           RTNSC                 Done
552                 ********************************************************************
553                 ********************************************************************
554                 **
555                 ** Name:    CLKUPD  -  Update Clock
556                 **
557                 ** Category:   TIME
558                 **
559                 ** Purpose:
560                 **      Set system clock to desired time.
```

```
561             **
562             **      This routine is the heart of the timekeeping system.
563             **      In addition to its obvious function of SETting a new
564             **      time, the code is used to perform routine clock
565             **      updates.
566             **
567             ** Entry:
568             **      R1 has time.
569             **      R0 has value of 512hz countdown timer corresponding
570             **        to time in R1.
571             **
572             ** Exit:
573             **      P=0.
574             **      Carry clear.
575             **      R1 still has time.
576             **      R0 has new value of 512hz countdown timer corresponding
577             **        to time in R1.
578             **
579             ** Calls:       GTFLAG, GETIRQ, GETPND, PUTPND, GETAF, MPY,
580             **              IDIV, PUTIRQ
581             **
582             ** Uses.......
583             **              A,B,C,D,D0,D1,P,R0,R1,S0-S11.
584             **
585             ** Stk lvls:    1
586             **
587             ** Detail:
588             **      Here is a brief summary of the clock systems workings:
589             **
590             **        The software has available to it one 24-bit 512hz
591             **      countdown timer.  For purposes of this discussion, a
592             **      "tick" is defined as 1/512 second.  The range of this
593             **      timer (in 2's complement arithmetic) is from 7FFFFFH
594             **      to -800000H ticks, or in decimal, 8388607 to -8388608.
595             **      This corresponds roughly to a range of 4.55 hours to
596             **      -4.55 hours.  The timer generates a service request
597             **      whenever it goes negative (high bit set).
598             **        Since the timer obviously cannot cover even a days
599             **      worth of ticks, let alone 9999 years worth, the clock
600             **      system uses the timer as an offset relative to a time
601             **      stored in RAM.  That is, it schedules a wakeup time
602             **      (called NXTIRQ), sets the timer to go negative (count
603             **      down through zero) at said time, and then can compute
604             **      the time on demand simply by computing:
605             **
606             **          TIME = NXTIRQ - (current timer value).
607             **
608             **        It is the CLKUPD routine which does the scheduling.
609             **      Calling CLKUPD says, in effect, "this is the current
610             **      time, schedule yourself to wake up."  Assuming we are
611             **      not in clock mode* and no alarms have been scheduled,
612             **      the routine simply schedules a wakeup for four hours
613             **      away.  When that happens, we wake up and perform
614             **      another clock update.  This goes on forever.
615             **
```

```
                **      { * Clock mode is a feature which allows lexfiles to
                **          implement a clock without having to reschedule an
                **          external alarm every second.  It is set by setting
                **          the flCLOC system flag and should be cleared when
                **          a clock mode is no longer desired.  In clock mode,
                **          the clock update interval is to the next 1-second
                **          boundary, and the exception flag is set (to force
                **          a poll) whenever the clock system is accessed }
                **
                **        If, however, any of the alarms (ALRM1-ALRM6) is set
                **      to go off less than 4 hours away, its time is used as
                **      the time-of-next-wakeup instead of 4 hours.
                **
                **        CLKUPD is executed whenever the time is computed
                **      (execution falls through from CMPT).  It is also
                **      executed when the time (time-of-day or date) is
                **      changed, although this occurs far less often.  Rather
                **      than simply taking "this is the current time" as input,
                **      the routine takes "this is the current time" and "this
                **      is what the timer read when it was computed".  That
                **      way any execution time taken after the time computation
                **      does not manifest itself as a time error.  This also
                **      serves to reduce the time-critical code to just a few
                **      lines at the end of the CLKUPD routine.
                **        When CLKUPD is passed a current time and timer value,
                **      it figures out the time-of-next-wakeup, stores that in
                **      NXTIRQ, and sets the timer appropriately.
                **
                ** Algorithm:
                **      {Time values use 12 nibbles.  Sometimes nibs 12-15
                **        are used for other things.  Registers specified
                **        without a subfield mean REG(0-11).}
                **      Get flCLOC flag.
                **      C = 4 hours (expressed in ticks).
                **      If we are not in clock mode, goto 1.
                **      C = Time to next second:
                **        (1 second) - [ (current time) mod (1 second) ] .
                **   1: {C now contains update interval if no alarms}
                **      A = current time (R1).
                **      D = time at update (A+C).
                **      B = current value of NXTIRQ.
                **  1a: Prepare to build bitmap in B[15-14] {bits 0-5 of that
                **        byte}.
                **      For D0 = ALRM6 downto ALRM1 do
                **        begin
                **          Read alarm time.
                **          If alarm time = 0 goto 3.
                **          If alarm time > current time goto 2.
                **          If alarm time < current value of NXTIRQ goto 3.
                **          Mark bitmap for this alarm {alarm is pending}.
                **          Goto 3.
                **       2: If alarm time > time-of-next-update {D} goto 3.
                **          D = alarm time.
                **       3:
                **      {D now contains the time-of-next-wakeup, which is
```

```
671             **        either the update time mentioned in "1:" above
672             **        or the earliest alarm time < 4 hours away.}
673             **        end
674             **      Move bitmap to B[B].
675             **      OR bitmap with PNDALM {new pending alarm bitmap}.
676             **      Save new bitmap in PNDALM and ST[B].
677             **      If alarms 0,1,2 or 5 are pending, set S12 {global
678             **        exception flag}.
679             **      If NXTIRQ < end-of-time then goto 4.
680             **      If current time < end-of-time then D=end-of-time;
681             **        goto 4 {schedule wakeup for end-of-time}.
682             **      current time := current time - end-of-time.
683             **      D = D - end-of-time.
684             **      B = 0.
685             **      Wrap TIMLAF and TIMLST around end-of-time.
686             **      Goto 1a.
687             **    4: {Beginning of accuracy factor correction code.  If
688             **        clock is running slow, we will want to schedule the
689             **        alarm early so as not to miss it.}
690             **      D=Time until new NXTIRQ {D-A}.
691             **      If AF<0 {if clock is fast} then goto 5.
692             **      D=D*AF/(AF+1) {compute reduced time to NXTIRQ}.
693             **      {End of AF code}
694             **    5: Add D to current time {R1}, store in NXTIRQ.
695             **      {D is the new value we want to store in the timer.
696             **        To compensate for any time taken since the time
697             **        indicated in R0/R1, we will actually store the value
698             **        [D-R0+(current timer)] in the timer.  Since
699             **        0 <= update amount <= 4 hours and the timer can
700             **        represent up to 4.55 hours + and -, only the
701             **        following things can screw us up:
702             **          (current timer)-R0 > .55 hours ... unlikely since
703             **                               current timer should be < R0.
704             **          (current timer)-R0 < -4.55 hours ... how long did
705             **                               processing take, anyway ?!?
706             **        If R0-(current timer) > D, an alarm came due during
707             **        processing and the timer will be negative.  We'll
708             **        catch it the next time through.}
709             **      C=D, CROEX {swap new timer value with old timer value}.
710             **      D=D-C {amount to add to current timer value}.
711             **      Watch timer until kerchunk {wait for full tick window}.
712             **      Read timer; add D; write timer {and timer enable bits
713             **        to control nibble}.
714             **      Recall current time {R1}.
715             **      Return.
716             **
717             ** History:
718             **
719             **     Date     Programmer            Modification
720             **    --------  ----------  ----------------------------------
721             **    06/07/82  NM          Added documentation
722             **
723             ****************************************************************
724             ****************************************************************
725 126C0 04    CLKUPD SETHEX
```

```
  726 126C2 20          P=     0
  727 126C4 3100        LC(2)  =flCLOC
  728 126C8 8E00        GOSUBL =GTFLAG       Get flag nib & mask for flCLOC
            00
  729 126CE 0E26        A=A&C  XS            AND flag nibble and mask
  730 126D2 78B1        GOSUB  LC708         4 hours (in 1/512 secs)
  731 126D6 BF2         CSL    W
  732 126D9 928         ?A=0   XS            Clock mode?
  733 126DC 81          GOYES  CLKU10        No, use 4 hour update
  734 126DE 3310        LCHEX  0001          Yes, find time to next second
            00
  735 126E4 111         A=R1                 Current time
  736 126E7 0E26        A=A&C  XS            Fractional second
  737 126EB 302         LCHEX  2
  738 126EE B32         C=C-A  X             Time until next second
  739 126F1 85C         ST=1   12            Set exception flag to force poll
  740 126F4 111   CLKU10 A=R1                Current time (1/512 sec)
  741 126F7 A72         C=C+A  W             Time at update
  742 126FA AF7         D=C    W             Hold for comparison
  743 126FD 73B2        GOSUB  GETIRQ        Read in old irq (1/512 sec)
  744 12701 AF5         B=C    W             Hold for comparison
  745 12704 2D    CLKU15 P=     13            For pending alarm bitmap to be...
  746 12706 3200        LCHEX  200           built in B[15-14]
            2
  747 1270B 1B55        DO=(5) =ALRM6        Point at last alarm & go back
            7F2
  748 12712 A92   CLKU20 C=0    WP
  749 12715 15EB        C=DATO 12            Read in alarm
  750 12719 91A         ?C=0   WP            Alarm time = 0?
  751 1271C E1          GOYES  CLKU40        Yes... ignore
  752 1271E 992         ?C>A   WP            Alarm after current time?
  753 12721 11          GOYES  CLKU30        Yes, consider for update
  754 12723 991         ?C<B   WP            No, alarm before oldirq?
  755 12726 41          GOYES  CLKU40        Yes, ignore
  756 12728 A92         C=0    WP            No, is pending alarm
  757 1272B 0E79        B=B!C  W             Mark bitmap
  758 1272F 5A0         GONC   CLKU40        B.E.T.
  759 12732 997   CLKU30 ?C>D  WP            Alarm before update?
  760 12735 50          GOYES  CLKU40        No, use update
  761 12737 A97         D=C    WP            Yes, use alarm
  762 1273A 18B   CLKU40 DO=DO- 12           Prev alarm
  763 1273D 81E         CSRB                 Shift bitmap bit
  764 12740 90A         ?C=0   P             Have we exhausted alarms?
  765 12743 FC          GOYES  CLKU20        No
  766 12745 09          C=ST
  767 12747 72F2        GOSUB  GETPND        Yes, read old pending bitmap
  768 1274B 811         BSLC                 Position new bitmap above it
  769 1274E 811         BSLC
  770 12751 0E6D        C=C!B  B             Or together for new pending alms
  771 12755 70F2        GOSUB  PUTPND
  772 12759 7F3E        GOSUB  SF12?         Save in status register & SF12?
  773 1275D 75A0        GOSUB  LC9999        C=TIME at end-of-time
  774 12761 9F7         ?D<C   W             New NXTIRQ < end-of-time?
  775 12764 63          GOYES  CLKU55        Yes, no action
  776 12766 9F2         ?A<C   W             Current time < end-of-time?
```

```
777 12769 E2          GOYES  CLKU54         Yes.  Wake up at end-of-time.
778 1276B B7A         A=A-C  W              No.  Wrap time around 0.
779 1276E 101         R1=A                  Store new time.
780 12771 B73         D=D-C  W              Wrap NXTIRQ around 0.
781 12774 AF1         B=0    W              Set old NXTIRQ to 0.
782 12777 AFA         A=C    W              Hold end-of-time
783 1277A 7282        GOSUB  GETLAF
784 1277E B72         C=C-A  W              Wrap TIMLAF around 0
785 12781 7592        GOSUB  PUTLAF
786 12785 7E92        GOSUB  GETLST
787 12789 B72         C=C-A  W              Wrap TIMLST around 0
788 1278C 72A2        GOSUB  PUTLST
789 12790 111         A=R1                  Recall current time
790 12793 607F        GOTO   CLKU15         Repeat scheduling process.
791 12797 AF7  CLKU54 D=C    W              NXTIRQ at end-of-time
792 1279A      CLKU55
793 1279A AF6         C=A    W              Current time
794 1279D B73         D=D-C  W              Time to nxtirq
795           * AF CORRECTION CODE
796           * WILL USE UPDATE OF TI=T/(1+1/AF)=AF*T/(AF+1) IF AF>0
797           * ELSE NO CORRECTION
798 127A0 7622        GOSUB  GETAF
799 127A4 412         GOC    CLKU60         No correction if AF<0
800 127A7 97A         ?C=0   W              AF=0?
801 127AA C1          GOYES  CLKU60         Yes, no correction
802 127AC AFA         A=C    W
803 127AF B76         C=C+1  W              AF+1
804 127B2 AFF         CDEX   W              T
805 127B5 7725        GOSUB  mpy            AF*t
806 127B9 AFB         C=D    W
807 127BC 7A15        GOSUB  idiv           AF*t/(AF+1)
808 127C0 AF6         C=A    W
809 127C3 AF7         D=C    W
810           * END OF AF CORRECTION CODE
811 127C6 119  CLKU60 C=R1                  Current time
812 127C9 A7B         C=C+D  W              Time at nextirq
813 127CC 7FE1        GOSUB  PUTIRQ
814 127D0 AFB         C=D    W              Time until NXTIRQ
815 127D3 128         CROEX                 Old timer value
816 127D6 B73         D=D-C  W              Amount to update timer
817 127D9 1B8F        D0=(5) =TIMER2        Timer to be updated
          2E2
818 127E0 25          P=     5
819 127E2 1524        A=DAT0 S
820 127E6 1564 CLKU70 C=DAT0 S
821 127EA 942         ?A=C   S
822 127ED 9F          GOYES  CLKU70         Loop until kerchunk
823 127EF 3200        LCHEX  800            Set timer enable bit
          8
824 127F4 15E5        C=DAT0 6              Plenty of time for update
825 127F8 A1B         C=C+D  WP
826 127FB 15C7        DAT0=C 8              Update done.
827 127FF 119         C=R1                  Recover current time
828 12802 20          P=     0
829 12804 03          RTNCC
```

```
                ****************************************************************
                ****************************************************************
                **
                ** Name:    LC9999  -  Load TIME At End-of-time
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Load TIME corresponding to 24:00:00, 31 Dec 9999.
                **
                ** Entry:
                **      None.
                **
                ** Exit:
                **      C=000092F2D17B0000.
                **      P=4.
                **      Carry unaffected.
                **
                ** Calls:      None.
                **
                ** Uses.......
                **  Exclusive: P,C.
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------   ---------------------------------
                **  06/10/82   NM           Wrote
                **
                ****************************************************************
                ****************************************************************
12806 AF2    LC9999 C=0    W
12809 24            P=     4
1280B 37B7          LCHEX  92F2D17B       C=000092F2D17B0000
      1D2F
      29
12815 01            RTN
                *-
                *-
12817 840    SOCNEG ST=0   0
1281A 500           RTNNC
1281D 850           ST=1   0
12820 BFA           C=-C   W
12823 02            RTNSC
```

```
874                 STITLE Low-level SET routines
875           ****************************************************************
876           ****************************************************************
877           **
878           ** Name:    SETIME  -  Set And Normal Adjust Routine
879           ** Name:(S) ADJN    -  Set And Normal Adjust Routine
880           **
881           ** Category:   TIME
882           **
883           ** Purpose:
884           **      Set new system time and keep track of error for
885           **      accuracy factor computation.
886           **
887           ** Entry:
888           **   SETIME, ADJN:
889           **      R1 = Current time (512ths sec since year 0).
890           **      R0 = Timer value corresponding to current time
891           **        (from CMPT).
892           **      R2 = New time to set (512ths sec since year 0).
893           **      HEX mode.
894           **
895           ** Exit:
896           **      R1=New time (R2 on entry).
897           **      R0=New timer value corresponding to time.
898           **      Carry clear.
899           **      P=0.
900           **
901           ** Calls:      CMPTE, GETOFS, PUTOFS, GETLST, PUTLST, GETLAF,
902           **             PUTLAF, CLKUPD (falls through)
903           **
904           ** Uses.......
905           **             A,B,C,D,P,D0,D1,R0,R1,S0-S11.
906           **
907           ** Stk lvls:   2
908           **
909           ** Detail:
910           **      SETIME, ADJN are two names for same entry point.
911           **
912           **      The adjustment amount is rounded to the nearest
913           **      half-hour.  The difference between that and the
914           **      adjustment amount (which will be between -15 and +15
915           **      minutes) is considered the error adjustment.  The rest
916           **      of the adjustment is considered a time zone change,
917           **      and is not added to TIMOFS (time error accumulator).
918           **
919           ** Algorithm:
920           **      Q := Newtime - currenttime {total adjustment amount}.
921           **      Te := sign(Q)*((abs(Q)+15) mod 30 - 15) {error
922           **        adjustment amount: between -15 and +15 minutes}.
923           **      TIMLST := TIMLST + Q - Te {update TIMLST by non-error
924           **        amount}.
925           **      TIMLAF := TIMLAF + Q {update TIMLAF by adjustment
926           **        amount}.
927           **      TIMOFS := TIMOFS + Te {update error accumulator by
928           **        error amount}.
```

```
                **      Fall through to CLKUPD.
                **
                ** History:
                **
                **     Date     Programmer            Modification
                **   --------   ----------   ---------------------------------
                **   06/08/82   NM           Added documentation
                **
                ********************************************************************
                ********************************************************************
12825           =SETIME
12825 112       =ADJN   A=R2                  New time
12828 119               C=R1                  Currenttime
1282B 101               R1=A                  Save for clkupd
1282E B7E               C=A-C   W             Q
12831 AF7               D=C     W
12834 501               GONC    SET10         Go if Q positive
12837 BFA               C=-C    W
1283A 7630              GOSUB   CMPTE         Compute Te
1283E BF9               B=-B    W
12841 6700              GOTO    SET20
12845 7B20  SET10       GOSUB   CMPTE
12849 7D31  SET20       GOSUB   GETOFS        Get error offset
1284D A79               C=C+B   W             Update
12850 7551              GOSUB   PUTOFS
12854 7FC1  SET30       GOSUB   GETLST
12858 A7B               C=C+D   W
1285B B79               C=C-B   W
1285E 70D1              GOSUB   PUTLST        Update timlst
12862 7A91              GOSUB   GETLAF
12866 A7B               C=C+D   W
12869 7DA1              GOSUB   PUTLAF        Update timlaf
1286D 85C               ST=1    12            Set Exception to force poll
12870 6F4E              GOTO    CLKUPD        Update clock
                ********************************************************************
                ********************************************************************
                **
                ** Name:    CMPTE   -  Compute Error Part Of Time Adjust
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Compute error correction component of time adjustment.
                **
                ** Entry:
                **      C = abs(adjustment amount).
                **      HEX mode.
                **
                ** Exit:
                **      B = Te/Sign(Q) (Te and Q explained in ADJN header).
                **
                ** Calls:      IDIV
                **
                ** Uses.......
                **              A,B,C,P
```

```
 984                ;**
 985                ;** Stk lvls:   1
 986                ;**
 987                ;** Algorithm:
 988                ;**      B := (abs(C)+15) mod 30 - 15.
 989                ;**
 990                ;** History:
 991                ;**
 992                ;**     Date     Programmer            Modification
 993                ;**   --------   ----------   --------------------------------
 994                ;**   06/08/82   NM           Added documentation
 995                ;**
 996                ;*****************************************************************
 997                ;*****************************************************************
 998 12874 AFA    CMPTE  A=C    W
 999 12877 7310          GOSUB  LC708          15 minutes
1000 1287B A7A           A=A+C  W
1001 1287E A76           C=C+C  W              30 minutes
1002 12881 7554          GOSUB  idiv           (abs(q)+15) mod 30 in b
1003 12885 7500          GOSUB  LC708          15 minutes
1004 12889 B71           B=B-C  W
1005 1288C 01            RTN
1006                ;*****************************************************************
1007                ;*****************************************************************
1008                ;**
1009                ;** Name:    LC708   -  Load A Constant
1010                ;**
1011                ;** Category:   LOCAL
1012                ;**
1013                ;** Purpose:
1014                ;**      Set C=0000000000070800, which is 15 minutes worth of
1015                ;**        ticks (tick=1/512th sec).
1016                ;**
1017                ;** Entry:
1018                ;**      None.
1019                ;**
1020                ;** Exit:
1021                ;**      C=0000000000070800.
1022                ;**      P=2.
1023                ;**
1024                ;** Calls:      None.
1025                ;**
1026                ;** Uses.......
1027                ;**             P,C.
1028                ;**
1029                ;** Stk lvls:   0
1030                ;**
1031                ;** History:
1032                ;**
1033                ;**    Date     Programmer            Modification
1034                ;**  --------   ----------   --------------------------------
1035                ;**  10/14/82   NM           Wrote.
1036                ;**
1037                ;*****************************************************************
1038                ;*****************************************************************
```

```
1039 1288E AF2   LC708  C=0    W
1040 12891 22           P=     2
1041 12893 3280         LCHEX  708
          7
1042 12898 03           RTNCC
1043            ****************************************************************
1044            ****************************************************************
1045            **
1046            ** Name:(S) ADJA    -  Absolute Time Adjust Routine
1047            **
1048            ** Category:   TIME
1049            **
1050            ** Purpose:
1051            **      Set new system time without timebase accuracy
1052            **      correction. The entire adjustment amount is considered
1053            **      a time zone change... none of it is an accuracy
1054            **      adjustment.
1055            **
1056            ** Entry:
1057            **      R1 = Current time (ticks since year 0).
1058            **      R0 = Timer value corresponding to current time
1059            **        (stored when CMPT was done) (ticks).
1060            **      R2 = New time to set (ticks since year 0).
1061            **      HEX mode.
1062            **
1063            ** Exit:
1064            **      R1=New time (R2 on entry).
1065            **      R0=New timer value corresponding to time.
1066            **      P=0.
1067            **      Carry clear.
1068            **
1069            ** Calls:      GETLST, PUTLST, GETLAF, PUTLAF, CLKUPD (falls
1070            **             through)
1071            **
1072            ** Uses.......
1073            **             A,B,C,D,P,D0,D1,R0,R1,S0-S11.
1074            **
1075            ** Stk lvls:   1
1076            **
1077            ** Algorithm:
1078            **      Q := Newtime - currenttime {total adjustment amount}.
1079            **      Te := 0  {error adjustment amount = 0}.
1080            **      TIMLST := TIMLST + Q - Te {update TIMLST by non-error
1081            **        amount}.
1082            **      TIMLAF := TIMLAF + Q {update TIMLAF by adjustment
1083            **        amount}.
1084            **      TIMOFS := TIMOFS + Te {update error accumulator by
1085            **        error amount}.
1086            **      Fall through to CLKUPD.
1087            **
1088            ** History:
1089            **
1090            **     Date     Programmer           Modification
1091            **   --------   ----------   ---------------------------------
1092            **   06/08/82   NM           Added documentation
```

```
1093                **
1094                ************************************************************************
1095                ************************************************************************
1096 1289A 112  =ADJA   A=R2
1097 1289D 119          C=R1
1098 128A0 101          R1=A
1099 128A3 B7E          C=A-C  W              Compute q
1100 128A6 AF7          D=C    W
1101 128A9 AF1          B=0    W              Te=0
1102 128AC 67AF         GOTO   SET30
1103                ************************************************************************
1104                ************************************************************************
1105                **
1106                ** Name:(S) EXACT   -  Compute New Accuracy Factor.
1107                **
1108                ** Category:   TIME
1109                **
1110                ** Purpose:
1111                **      Inform time system that time currently contained is
1112                **      exact.
1113                **
1114                **      The first time EXACT is called after a coldstart or
1115                **      a RESET CLOCK, the exact flag is clear.
1116                **      This routine will simply set it, note the current time
1117                **      and start a new adjustment period.
1118                **        Each subsequent call will note the elapsed time since
1119                **      the last call and the corrections which have been
1120                **      applied since the last call.  From this an accuracy
1121                **      factor is computed.
1122                **
1123                ** Entry:
1124                **      None.
1125                **
1126                ** Exit:
1127                **      A new adjustment period has been started.
1128                **      Carry set: Reasonable accuracy factor computed.
1129                **      Carry clear: Illegal accuracy factor computed.
1130                **
1131                ** Calls:      CMPT, GTFLAG, COMPAF, PUTAF, PUTOFS
1132                **             PUTLST.
1133                **
1134                ** Uses.......
1135                **             A,B,C,D,P,D0,D1,R0,R1,S0-S11.
1136                **
1137                ** Stk lvls:   2
1138                **
1139                ** Algorithm:
1140                **      If exact=true then
1141                **        compute AF
1142                **        if AF valid then store AF.
1143                **      TIMLST:=TIMLAF:=currenttime  {start of new adjustment
1144                **        period}.
1145                **      TIMOFS:=0.
1146                **      EXACT:=true.
1147                **      return with carry clear if:
```

```
1148            **        exact was false
1149            **        exact was true, computed AF is valid.
1150            **      return with carry set if:
1151            **        exact was true, computed AF was invalid.
1152            **
1153            ** History:
1154            **
1155            **     Date     Programmer            Modification
1156            **   --------   ----------   ---------------------------------
1157            **   06/08/82   NM           Added documentation
1158            **
1159            ********************************************************************
1160            ********************************************************************
1161 128B0 7EFC =EXACT  GOSUB  CMPT
1162 128B4 3100         LC(2)  =flEXAC
1163 128B8 8E00         GOSUBL =GTFLAG        Get flag nib & mask for flEXAC
         00
1164 128BE AA8          B=A    XS
1165 128C1 0E21         B=B&C  XS             AND flag nibble with mask
1166 128C5 92D          ?B#0   XS             EXACT flag set?
1167 128C8 D0           GOYES  EXAC10         Yes
1168 128CA 0E2E         A=A!C  XS             No, set it.
1169 128CE 1502         DAT0=A XS             Write out new flag nibble
1170 128D2 561          GONC   EXAC30         New adjustment period.  B.E.T.
1171 128D5 734D  EXAC10 GOSUB  COMPAF         Compute accuracy factor
1172 128D9 480          GOC    EXAC20         Go if reasonable
1173 128DC 7900         GOSUB  EXAC30         Unreasonable.  new adj. period.
1174 128E0 02           RTNSC
1175 128E2 AF6   EXAC20 C=A    W
1176 128E5 7A01         GOSUB  PUTAF          Store new AF
1177 128E9 79B0  EXAC30 GOSUB  PUTOF0         Reset timofs
1178 128ED 119          C=R1                  Current time
1179 128F0 7E31         GOSUB  PUTLST         Reset time of last exact
1180 128F4 6521         GOTO   PUTLAF         Reset time of last AF adj
1181            ********************************************************************
1182            ********************************************************************
1183            **
1184            ** Name:    RESETC  -  Clear AF And Exact Flag.
1185            **
1186            ** Category:   TIME
1187            **
1188            ** Purpose:
1189            **      Clear exact flag and set AF = 0.
1190            **
1191            ** Entry:
1192            **      None.
1193            **
1194            ** Exit:
1195            **      P=0.
1196            **      Carry clear.
1197            **      flEXAC has been cleared.
1198            **      (TIMAF) = 000000.
1199            **
1200            ** Calls:      PUTAF, SFLAGC (falls through).
1201            **
```

```
1202            ** Uses.......
1203            **              A,B,C,D,D0,D1,P.
1204            **
1205            ** Stk lvls:   2
1206            **
1207            ** Algorithm:
1208            **      Clear system exact flag.
1209            **      AF := 0.
1210            **
1211            ** History:
1212            **
1213            **     Date     Programmer            Modification
1214            **   --------   ----------   ---------------------------------
1215            **   06/08/82   NM           Added documentation
1216            **
1217            ********************************************************************
1218            ********************************************************************
1219 128F8      =RESETC
1220 128F8 04           SETHEX
1221 128FA AF2          C=0     W
1222 128FD 72F0         GOSUB   PUTAF           Clear accuracy factor
1223 12901 20           P=      0
1224 12903 3100         LC(2)   =f1EXAC
1225 12907 8C00         GOLONG  =SFLAGC         Clear EXACT flag
          00
1226            ********************************************************************
1227            ********************************************************************
1228            **
1229            ** Name:(S) SETALM  -  Set Absolute Alarm Time
1230            **
1231            ** Category:   TIME
1232            **
1233            ** Purpose:
1234            **      Set detonation time for any of alarms 1-6.
1235            **
1236            ** Entry:
1237            **      Alarm time in A[11-0] (ticks since 1 Jan 0000).
1238            **      Alarm#-1 (0-5) in C[0].
1239            **
1240            ** Exit:
1241            **      Through CMPT.
1242            **      Carry clear.
1243            **      P=0.
1244            **      R1 = Current time (512ths sec since year 0)
1245            **      R0 = timer value corresponding to current time.
1246            **
1247            ** Calls:      GETPND, PUTPND, CMPT (falls through).
1248            **
1249            ** Uses.......
1250            **              A,B,C,D,P,D0,D1,S0-S11,R0,R1.
1251            **
1252            ** Stk lvls:   2
1253            **
1254            ** Algorithm:
1255            **      Write alarm time to proper RAM location (ALRM1-ALRM6).
```

```
1256                ;**      Clear proper bit (0-5) in PNDALM.
1257                ;**      Fall through to CMPT.
1258                ;**
1259                ;** History:
1260                ;**
1261                ;**     Date     Programmer          Modification
1262                ;**    --------  ----------  --------------------------------
1263                ;**    06/09/82  NM          Added documentation
1264                ;**
1265                ;*****************************************************************
1266                ;*****************************************************************
1267 1290D 04     =SETALM SETHEX
1268 1290F 7F30           GOSUB  PUTALM         Store this alarm
1269 12913 6E9C           GOTO   CMPT           Read time and update
1270                ;*****************************************************************
1271                ;*****************************************************************
1272                ;**
1273                ;** Name:(S) SETALR  -  Set Alarm Relative To Current Time
1274                ;**
1275                ;** Category:   TIME
1276                ;**
1277                ;** Purpose:
1278                ;**      Set alarm time relative to current time.
1279                ;**
1280                ;** Entry:
1281                ;**      A[11-0] = Interval (512ths sec)
1282                ;**      C[0] = Alarm#-1
1283                ;**
1284                ;** Exit:
1285                ;**      Through CLKUPD.
1286                ;**      Carry clear.
1287                ;**      P=0.
1288                ;**      R1 = current time (512ths sec since year 0).
1289                ;**      R0 = timer value corresponding to current time.
1290                ;**
1291                ;** Calls:      CMPT, SETALM (falls through).
1292                ;**
1293                ;** Uses.......
1294                ;**             A,B,C,D,P,D0,D1,R0,R1,R3,S0-S11.
1295                ;**
1296                ;** Stk lvls:   2
1297                ;**
1298                ;** Algorithm:
1299                ;**      Add interval to current time.
1300                ;**      Wrap around end-of-time.
1301                ;**      Write out new alarm time to appropriate slot.
1302                ;**      Update clock.
1303                ;**
1304                ;** History:
1305                ;**
1306                ;**     Date     Programmer          Modification
1307                ;**    --------  ----------  --------------------------------
1308                ;**    06/09/82  NM          Added documentation
1309                ;**
1310                ;*****************************************************************
```

```
1311            ****************************************************************
1312 12917 816 =SETALR CSRC                    Alarm# to C[S]
1313 1291A ACA         A=C     S
1314 1291D 103         R3=A                    Save alarm information
1315 12920 7E8C        GOSUB   CMPT            Compute current time
1316 12924 113         A=R3                    Recall alarm information
1317 12927 2E          P=      14
1318 12929 A1A         A=A+C   WP              Compute alarm time
1319 1292C ACE         ACEX    S
1320 1292F AC7  SETA10 D=C     S               Stash alarm#
1321 12932 70DE        GOSUB   LC9999          Load end-of-time
1322 12936 70A3        GOSUB   idiv            Compute MOD.
1323 1293A AF4         A=B     W               (alarm time) mod (end-of-time)
1324 1293D 97C         ?A#0    W               New alarm time=0?
1325 12940 40          GOYES   SETA20          No
1326 12942 E4          A=A+1   A               Intentional change so not ignore
1327 12944 ACB  SETA20 C=D     S               Recover alarm#
1328 12947 812         CSLC                    Alarm#-1 to C[0]
1329 1294A 7400        GOSUB   PUTALM          Set alarm
1330 1294E 617D        GOTO    CLKUPD          Update time system
1331            ****************************************************************
1332            ****************************************************************
1333            **
1334            ** Name:    PUTALM  -  Store Alarm In Requested Alarm#
1335            **
1336            ** Category:   TIME
1337            **
1338            ** Purpose:
1339            **      Store alarm in slot for requested alarm#.  Also clear
1340            **      pending alarm bit for requested alarm#.
1341            **
1342            ** Entry:
1343            **      C[0] = Alarm#-1.
1344            **      A[11-0] = Alarm time to store.
1345            **
1346            ** Exit:
1347            **      Carry clear: New alarm time has been stored for
1348            **        requested alarm#.  Pending alarm bit has been
1349            **        cleared.
1350            **      Carry set: Requested illegal alarm#.
1351            **      P=0.
1352            **
1353            ** Calls:      GETPND, PUTPND.
1354            **
1355            ** Uses.......
1356            **             B[A],C[A],P,D1.
1357            **
1358            ** Stk lvls:   1
1359            **
1360            ** History:
1361            **
1362            **    Date      Programmer              Modification
1363            **  --------    ----------   -----------------------------------
1364            **  07/15/82    NM           Extracted from other routines
1365            **
```

```
1366            ********************************************************************
1367            ********************************************************************
1368 12952 20  =PUTALM P=     0
1369 12954 D1          B=0    A
1370 12956 E5          B=B+1  A              For locating this alarm bit
1371 12958 1F91        D1=(5) =ALRM1
           7F2
1372 1295F A0E  PUTA10 C=C-1  P              Loop to locate desired alarm
1373 12962 4C0         GOC    PUTA20         Go if found
1374 12965 17B         D1=D1+ 12             Point to next alarm
1375 12968 C5          B=B+B  A              Shift bit for bitmap
1376 1296A 54F         GONC   PUTA10         B.E.T. if alarm#<8
1377 1296D 02          RTNSC                 Return if illegal alarm#
1378 1296F 3102 PUTA20 LCHEX  20             Max allowable alarm bit
1379 12973 9E1         ?B>C   B              Legal alarm #?
1380 12976 00          RTNYES                No
1381 12978 159B        DAT1=A 12             Write out alarm time
1382 1297C 7DB0        GOSUB  GETPND         Get PNDALM bitmap
1383 12980 FD          B=-B-1 A              Zero bit for this alarm
1384 12982 0EF5        C=C&B  A              Clear pending bit
1385 12986 62C0        GOTO   PUTPND
```

```
1386                    STITLE Store and Recall routines
1387                ********************************************************************
1388                ********************************************************************
1389                **
1390                ** Name:    GETOFS  -  Read TIMOFS From RAM
1391                **
1392                ** Category:   LOCAL
1393                **
1394                ** Purpose:
1395                **      Read TIMOFS from RAM.
1396                **
1397                ** Entry:
1398                **      HEX mode.
1399                **
1400                ** Exit:
1401                **      C = (TIMOFS).
1402                **      P=11.
1403                **      Carry set iff quantity negative.
1404                **
1405                ** Calls:      None.
1406                **
1407                ** Uses.......
1408                **              C,P,D1
1409                **
1410                ** Stk lvls:   0
1411                **
1412                ** Detail:
1413                **      TIMOFS = Accumulated error amount for AF computation
1414                **        (48-bit 2's-complement integer,
1415                **         range = [ -8710.53 , 8710.53 ] years).
1416                **
1417                ** History:
1418                **
1419                **    Date     Programmer              Modification
1420                **  --------   ----------   ---------------------------------
1421                **  06/09/82   NM           Added documentation
1422                **
1423                ********************************************************************
1424                ********************************************************************
1425 1298A 1F36  GETOFS D1=(5) =TIMOFS
               7F2
1426 12991 AF2          C=0    W
1427 12994 15FB         C=DAT1 12
1428 12998 80DB         P=C    11
1429 1299C 80FF         CPEX   15
1430 129A0 2B           P=     11
1431 129A2 6F30         GOTO   GETA10          Else do sign extend
1432                ********************************************************************
1433                ********************************************************************
1434                **
1435                ** Name:    PUTOFS  -  Store TIMOFS Into RAM
1436                **
1437                ** Category:   LOCAL
1438                **
1439                ** Purpose:
```

```
1440                **      Store new value of (TIMOFS) -- error accumulator.
1441                **
1442                ** Entry:
1443                **      C = new value.
1444                **
1445                ** Exit:
1446                **      Value has been stored in (TIMOFS).
1447                **      Carry clear.
1448                **
1449                ** Calls:      None.
1450                **
1451                ** Uses.......
1452                **              D1.
1453                **
1454                ** Stk lvls:   0
1455                **
1456                ** Detail:
1457                **      Inverse of GETOFS.
1458                **
1459                ** History:
1460                **
1461                **    Date      Programmer              Modification
1462                **  --------   ----------   ----------------------------------
1463                **  06/09/82   NM           Added documentation
1464                **
1465                ********************************************************************
1466                ********************************************************************
1467 129A6 AF2     PUTOFO C=0    W
1468 129A9 1F36    PUTOFS D1=(5) =TIMOFS
           7F2
1469 129B0 6070            GOTO   PUTL12
1470                ********************************************************************
1471                ********************************************************************
1472                **
1473                ** Name:    GETIRQ  -  Read NXTIRQ From RAM.
1474                **
1475                ** Category:   LOCAL
1476                **
1477                ** Purpose:
1478                **      Read NXTIRQ from RAM.
1479                **      NXTIRQ is the time of next wakeup (512ths sec).
1480                **        (48-bit unsigned integer.  Range=[ 0 , 17421] years).
1481                **
1482                ** Entry:
1483                **      HEX mode.
1484                **
1485                ** Exit:
1486                **      C = (NXTIRQ).
1487                **      P=11.
1488                **
1489                ** Calls:      None.
1490                **
1491                ** Uses.......
1492                **              C,P,D1.
1493                **
```

```
1494                ** Stk lvls:   0
1495                **
1496                ** History:
1497                **
1498                **    Date      Programmer              Modification
1499                **  --------   ----------   ----------------------------------
1500                **  06/09/82   NM           Added documentation
1501                **
1502                ********************************************************************
1503                ********************************************************************
1504 129B4 1FD0   GETIRQ D1=(5) =NXTIRQ
             7F2
1505 129BB 6B40           GOTO   GETL12
1506                ********************************************************************
1507                ********************************************************************
1508                **
1509                ** Name:    PUTIRQ  -  Write NXTIRQ To RAM.
1510                **
1511                ** Category:   LOCAL
1512                **
1513                ** Purpose:
1514                **      Write new value of (NXTIRQ).
1515                **
1516                ** Entry:
1517                **      C[11-0] = new value.
1518                **
1519                ** Exit:
1520                **      Value has been written to (NXTIRQ).
1521                **      Carry clear.
1522                **
1523                ** Calls:      None.
1524                **
1525                ** Uses.......
1526                **             D1.
1527                **
1528                ** Stk lvls:   0
1529                **
1530                ** History:
1531                **
1532                **    Date      Programmer              Modification
1533                **  --------   ----------   ----------------------------------
1534                **  06/09/82   NM           Added documentation
1535                **
1536                ********************************************************************
1537                ********************************************************************
1538 129BF 1FD0   PUTIRQ D1=(5) =NXTIRQ
             7F2
1539 129C6 6A50           GOTO   PUTL12
1540                ********************************************************************
1541                ********************************************************************
1542                **
1543                ** Name:    GETAF   -  Read Adjustment Factor From RAM
1544                **
1545                ** Category:   LOCAL
1546                **
```

```
1547                ** Purpose:
1548                **      Read AF from RAM.
1549                **
1550                ** Entry:
1551                **      HEX mode.
1552                **
1553                ** Exit:
1554                **      C = (TIMAF), sign extended to full word.
1555                **      Carry set iff negative.
1556                **
1557                ** Calls:      None.
1558                **
1559                ** Uses.......
1560                **             D1,C.
1561                **
1562                ** Stk lvls:   0
1563                **
1564                **
1565                ** Detail:
1566                **      Adjustment factor = #ticks to wait before adding a tick
1567                **        (24-bit 2's complement integer).
1568                **        Positive AF corrects a slow clock;
1569                **        Negative AF corrects a fast clock.
1570                **
1571                ** History:
1572                **
1573                **    Date      Programmer              Modification
1574                **  --------   ----------   ----------------------------------
1575                **  06/10/82   NM           Added Documentation
1576                **
1577                ********************************************************************
1578                ********************************************************************
1579 129CA 1F78  GETAF  D1=(5) =TIMAF
                    7F2
1580 129D1 AF2          C=0    W
1581 129D4 15F5         C=DAT1 6              Accuracy factor (secs/sec)
1582 129D8 80D5         P=C    5
1583 129DC 80FF         CPEX   15             Copy sign nib to nib 15
1584 129E0 25           P=     5
1585 129E2 A46   GETA10 C=C+C  S
1586 129E5 AC2          C=0    S
1587 129E8 500   GETA20 RTNNC                 Hibit clear implies positive
1588 129EB B9A          C=-C   WP              Perform sign extend
1589 129EE BFA          C=-C   W
1590 129F1 02           RTNSC
1591                ********************************************************************
1592                ********************************************************************
1593                **
1594                ** Name:    PUTAF   -  Store Adjustment Factor Into RAM
1595                **
1596                ** Category:   LOCAL
1597                **
1598                ** Purpose:
1599                **      Store AF into RAM.
1600                **
```

```
1601                ** Entry:
1602                **      C[5-0] = AF.
1603                **
1604                ** Exit:
1605                **      Carry unaffected
1606                **
1607                ** Calls:      None.
1608                **
1609                ** Uses.......
1610                **              D1.
1611                **
1612                ** Stk lvls:   0
1613                **
1614                ** History:
1615                **
1616                **    Date      Programmer              Modification
1617                **  --------    ----------   ----------------------------------
1618                **  06/10/82    NM           Added documentation
1619                **
1620                ********************************************************************
1621                ********************************************************************
1622 129F3 1F78  PUTAF  D1=(5) =TIMAF
           7F2
1623 129FA 15D5         DAT1=C 6
1624 129FE 01           RTN
1625                ********************************************************************
1626                ********************************************************************
1627                **
1628                ** Name:    GETLAF  -  Read TIMLAF From RAM
1629                **
1630                ** Category:   LOCAL
1631                **
1632                ** Purpose:
1633                **      Read TIMLAF from RAM.
1634                **      TIMLAF is the time of last AF adjustment
1635                **        (48-bit integer, usually unsigned).
1636                **
1637                ** Entry:
1638                **      HEX mode.
1639                **
1640                ** Exit:
1641                **      C = (TIMLAF).
1642                **      P=11.
1643                **
1644                ** Calls:      None.
1645                **
1646                ** Uses.......
1647                **              D1,P,C.
1648                **
1649                ** Stk lvls:   0
1650                **
1651                ** History:
1652                **
1653                **    Date      Programmer              Modification
1654                **  --------    ----------   ----------------------------------
```

```
1655            ** 06/10/82   NM          Added documentation
1656            **
1657            ********************************************************************
1658            ********************************************************************
1659 12A00 1FB7  GETLAF D1=(5) =TIMLAF
           7F2
1660            *
1661            * Following is the common code for the GET routines for
1662            * retrieving TIMLAF, TIMLST, NXTIRQ.  All of these are 48-bit
1663            * unsigned quantities which can range from 0 to 92F2D17AFFFF.
1664            *
1665            * Because of the peculiarities of adjustment factors, it is
1666            * possible for TIMLAF actually to be slightly negative under
1667            * certain circumstances.  This code performs a sign extend
1668            * only if the high nibble of the requested quantity if F.
1669            *
1670 12A07 AF2   GETL12 C=0    W
1671 12A0A 15FB         C=DAT1 12
1672 12A0E 2B           P=     11
1673 12A10 B06          C=C+1  P
1674 12A13 A0E          C=C-1  P              Set carry if C[11]=F
1675 12A16 61DF         GOTO   GETA20         Do conditional sign extend
1676            ********************************************************************
1677            ********************************************************************
1678            **
1679            ** Name:    PUTLAF  -  Store TIMLAF In RAM
1680            **
1681            ** Category:   LOCAL
1682            **
1683            ** Purpose:
1684            **      Store TIMLAF in RAM.
1685            **
1686            ** Entry:
1687            **      C[11-0] = new value to store.
1688            **
1689            ** Exit:
1690            **      Value has been stored in TIMLAF.
1691            **      Carry clear.
1692            **
1693            ** Calls:      None.
1694            **
1695            ** Uses.......
1696            **              D1.
1697            **
1698            ** Stk lvls:   0
1699            **
1700            ** History:
1701            **
1702            **    Date      Programmer              Modification
1703            **  --------    ----------   ----------------------------------
1704            **  06/11/82    NM           Added documentation
1705            **
1706            ********************************************************************
1707            ********************************************************************
1708 12A1A 1FB7  PUTLAF D1=(5) =TIMLAF
```

```
            7F2
 1709 12A21 15DB  PUTL12 DAT1=C 12
 1710 12A25 03           RTNCC
 1711               ********************************************************************
 1712               ********************************************************************
 1713               **
 1714               ** Name:    GETLST  -  Read TIMLST From RAM
 1715               **
 1716               ** Category:   LOCAL
 1717               **
 1718               ** Purpose:
 1719               **      Read (TIMLST) from RAM.
 1720               **      (TIMLST) is time of last exact.
 1721               **
 1722               ** Entry:
 1723               **      None.
 1724               **
 1725               ** Exit:
 1726               **      C=(TIMLST).
 1727               **      P=11.
 1728               **
 1729               ** Calls:      None.
 1730               **
 1731               ** Uses.......
 1732               **             D1,P,C.
 1733               **
 1734               ** Stk lvls:   0
 1735               **
 1736               ** History:
 1737               **
 1738               **    Date     Programmer              Modification
 1739               **  --------   ----------   ------------------------------------
 1740               **  06/11/82   NM           Added documentation
 1741               **
 1742               ********************************************************************
 1743               ********************************************************************
 1744 12A27 1FF6  GETLST D1=(5) =TIMLST
            7F2
 1745 12A2E 68DF         GOTO   GETL12
 1746               ********************************************************************
 1747               ********************************************************************
 1748               **
 1749               ** Name:    PUTLST  -  Store TIMLST In RAM
 1750               **
 1751               ** Category:   LOCAL
 1752               **
 1753               ** Purpose:
 1754               **      Store TIMLST into RAM.
 1755               **
 1756               ** Entry:
 1757               **      C = new value to store.
 1758               **
 1759               ** Exit:
 1760               **      Value has been stored in TIMLST.
 1761               **
```

```
1762            ** Calls:      None.
1763            **
1764            ** Uses.......
1765            **             D1.
1766            **
1767            ** Stk lvls:   0
1768            **
1769            ** History:
1770            **
1771            **    Date     Programmer              Modification
1772            **  --------   ----------   ----------------------------------
1773            **  06/11/82   NM           Added documentation
1774            **
1775            ********************************************************************
1776            ********************************************************************
1777 12A32 1FF6  PUTLST D1=(5) =TIMLST
         7F2
1778 12A39 67EF         GOTO   PUTL12
1779            ********************************************************************
1780            ********************************************************************
1781            **
1782            ** Name:    GETPND  -  Read PNDALM From RAM
1783            **
1784            ** Category:   LOCAL
1785            **
1786            ** Purpose:
1787            **      Read PNDALM from RAM.
1788            **
1789            ** Entry:
1790            **      None.
1791            **
1792            ** Exit:
1793            **      C[B] = (PNDALM).
1794            **      Carry Clear.
1795            **
1796            ** Calls:      None.
1797            **
1798            ** Uses.......
1799            **             D1,C[B].
1800            **
1801            ** Stk lvls:   0
1802            **
1803            ** Detail:
1804            **      PNDALM is bitmap of pending alarms:
1805            **        Bit 0: Alarm #1--On timer#1
1806            **        Bit 1: Alarm #2--On timer#2
1807            **        Bit 2: Alarm #3--On timer#3
1808            **        Bit 3: Alarm #4--Timeout
1809            **        Bit 4: Alarm #5--Pause
1810            **        Bit 5: Alarm #6--External alarm
1811            **
1812            ** History:
1813            **
1814            **    Date     Programmer              Modification
1815            **  --------   ----------   ----------------------------------
```

```
1816                **  06/11/82   NM          Added documentation
1817                **
1818                ********************************************************************
1819                ********************************************************************
1820 12A3D 1F16 7F2  GETPND D1=(5) =PNDALM
1821 12A44 14F       C=DAT1 B
1822 12A47 03        RTNCC
1823                ********************************************************************
1824                ********************************************************************
1825                **
1826                ** Name:    PUTPND  -  Write PNDALM To RAM
1827                **
1828                ** Category:   GENUTL
1829                **
1830                ** Purpose:
1831                **      Write PNDALM to RAM.
1832                **
1833                ** Entry:
1834                **      C[B] = new value to store.
1835                **
1836                ** Exit:
1837                **      (PNDALM) = C[B] on entry.
1838                **      Carry Clear.
1839                **
1840                ** Calls:      None.
1841                **
1842                ** Uses.......
1843                **              D1.
1844                **
1845                ** Stk lvls:   0
1846                **
1847                ** History:
1848                **
1849                **    Date     Programmer              Modification
1850                **  --------   ----------   ---------------------------------
1851                **  06/11/82   NM           Added documentation
1852                **
1853                ********************************************************************
1854                ********************************************************************
1855 12A49 1F16 7F2 =PUTPND D1=(5) =PNDALM
1856 12A50 14D       DAT1=C B
1857 12A53 03        RTNCC
```

```
                        STITLE Non-TIME utilities
12A55 8D00 =FLOATA GOVLNG =FLOAT
      000
*****************************************************************
*****************************************************************
**
** Name:    STD0D1  -  Stash D0 And D1 In R4.
**
** Category:   LOCAL
**
** Purpose:
**      Save D0 and D1 in R4.
**
** Entry:
**      None.
**
** Exit:
**      Carry clear.
**      D0, D1 preserved.
**      R4[9-5]=D1.
**      R4[14-10]=D0.
**
** Calls:      None.
**
** Uses.......
**              A,R4.
**
** Stk lvls:   0
**
** History:
**
**    Date      Programmer              Modification
**  --------    ----------   ----------------------------------
**  06/11/82    NM           Added documentation
**
*****************************************************************
*****************************************************************
12A5C 132   STD0D1 A0EX
12A5F BF0          ASL    W
12A62 BF0          ASL    W
12A65 BF0          ASL    W
12A68 BF0          ASL    W
12A6B BF0          ASL    W
12A6E 133          A1EX
12A71 BF0          ASL    W
12A74 BF0          ASL    W
12A77 BF0          ASL    W
12A7A BF0          ASL    W
12A7D BF0          ASL    W
12A80 104          R4=A                    Stash D0,D1.  Fall through...
*****************************************************************
*****************************************************************
**
** Name:    RCD0D1  -  Restore D0 And D1
**
```

```
1912                ** Category:   LOCAL
1913                **
1914                ** Purpose:
1915                **      Restore D0 and D1 from R4.
1916                **
1917                ** Entry:
1918                **      R4[14-10] = previous D0 stored by STD0D1.
1919                **      R4[9-5] = previous D1 stored by STD0D1.
1920                **
1921                ** Exit:
1922                **      D0 = R4[14-10].
1923                **      D1 = R4[9-5].
1924                **      Carry clear.
1925                **
1926                ** Calls:      None.
1927                **
1928                ** Uses.......
1929                **              A,D0,D1.
1930                **
1931                ** Stk lvls:   0
1932                **
1933                ** History:
1934                **
1935                **    Date      Programmer              Modification
1936                **  --------    ----------   -----------------------------------
1937                **  06/11/82    NM           Added documentation
1938                **
1939                ********************************************************************
1940                ********************************************************************
1941                *
1942                * NOTE: Above code falls into this
1943                *
1944 12A83 114    RCD0D1 A=R4
1945 12A86 BF4          ASR    W
1946 12A89 BF4          ASR    W
1947 12A8C BF4          ASR    W
1948 12A8F BF4          ASR    W
1949 12A92 BF4          ASR    W
1950 12A95 133          AD1EX
1951 12A98 BF4          ASR    W
1952 12A9B BF4          ASR    W
1953 12A9E BF4          ASR    W
1954 12AA1 BF4          ASR    W
1955 12AA4 BF4          ASR    W
1956 12AA7 132          AD0EX
1957 12AAA 03           RTNCC
1958                ********************************************************************
1959                ********************************************************************
1960                **
1961                ** Name:     ST01     -  Save Scratch Regs In RAM
1962                **
1963                ** Category:   SAVUTL
1964                **
1965                ** Purpose:
1966                **      Save R0,R1 in RAM.
```

```
1967                **
1968                ** Entry:
1969                **      None.
1970                **
1971                ** Exit:
1972                **      SCRTCH[0-15] = R0.
1973                **      SCRTCH[16-31] = R1.
1974                **      Carry clear.
1975                **
1976                ** Calls:      None.
1977                **
1978                ** Uses.......
1979                **              D0,A, SCRTCH[0-31].
1980                **
1981                ** Stk lvls:   0
1982                **
1983                ** History:
1984                **
1985                **    Date      Programmer              Modification
1986                **  --------    ----------   ----------------------------------
1987                **  06/11/82    NM           Added documentation
1988                **
1989                ********************************************************************
1990                ********************************************************************
1991 12AAC 1B10 =ST01   D0=(5) =SCRTCH
           9F2
1992 12AB3 110          A=R0
1993 12AB6 1507         DAT0=A W
1994 12ABA 16F          D0=D0+ 16
1995 12ABD 111          A=R1
1996 12AC0 1507         DAT0=A W
1997 12AC4 03           RTNCC
1998                ********************************************************************
1999                ********************************************************************
2000                **
2001                ** Name:    RC01    -  Recall Scratch Regs From RAM
2002                **
2003                ** Category:   SAVUTL
2004                **
2005                ** Purpose:
2006                **      Restore R0,R1 from RAM.
2007                **
2008                ** Entry:
2009                **      SCRTCH[0-31] contain R0, R1 saved by ST01.
2010                **      None.
2011                **
2012                ** Exit:
2013                **      R0,R1 restored to values in SCRTCH.
2014                **      P=0.
2015                **      Carry set.
2016                **
2017                ** Calls:      None.
2018                **
2019                ** Uses.......
2020                **              D0,A,R0,R1,P.
```

```
2021                **
2022                ** Stk lvls:   0
2023                **
2024                ** History:
2025                **
2026                **    Date      Programmer            Modification
2027                **  --------    ----------   ------------------------------------
2028                **  06/11/82    NM           Added documentation
2029                **
2030                ********************************************************************
2031                ********************************************************************
2032 12AC6 1B10 =RC01    D0=(5) =SCRTCH
           9F2
2033 12ACD 1527          A=DAT0 W
2034 12AD1 100           R0=A
2035 12AD4 16F           D0=D0+ 16
2036 12AD7 1527          A=DAT0 W
2037 12ADB 101           R1=A
2038 12ADE 20            P=     0
2039 12AE0 02            RTNSC
2040                ********************************************************************
2041                ********************************************************************
2042                **
2043                ** Name:(S) RJUST   -  Unfloat A Floating-Point Number
2044                **
2045                ** Category:   CONVRT
2046                **
2047                ** Purpose:
2048                **      Unfloat a 12-digit form floating-point number.
2049                **
2050                ** Entry:
2051                **      12-digit floating-point number in A
2052                **        (sign ignored).
2053                **
2054                ** Exit:
2055                **      Error exit (Inv Arg) if NaN passed.
2056                **      A[W] = Right-justified decimal integer version of
2057                **        argument.
2058                **      Carry set: Arg was infinity; result=1E16 - 1.
2059                **      Carry clear: Arg was finite; arg >= 1E16 returned
2060                **        as 1E16 - 1.
2061                **      DEC mode.
2062                **      P=0.
2063                **
2064                ** Calls:      None.
2065                **
2066                ** Uses.......
2067                **              A,C,P.
2068                **
2069                ** Stk lvls:   0
2070                **
2071                ** Detail:
2072                **      Input: 0123000000000002    Output: 0000000000000123
2073                **      Input: 0123500000000002    Output: 0000000000000124
2074                **      Input: 0123456789870007    Output: 0000000012345679
```

```
2075                **      Input: 0987000000000998    Output: 0000000000000000
2076                **      Input: 0987000000000050    Output: 9999999999999999
2077                **
2078                ** History:
2079                **
2080                **    Date      Programmer              Modification
2081                **  --------    ----------   ----------------------------------
2082                **  06/18/82    NM           Added documentation
2083                **
2084                ********************************************************************
2085                ********************************************************************
2086 12AE2 05    =RJUST  SETDEC
2087 12AE4 20            P=      0
2088 12AE6 3200          LCHEX   F00
           F
2089 12AEB 926           ?A#C    XS              Inf or nan?
2090 12AEE 31            GOYES   RJUS07          No
2091 12AF0 968           ?A=0    B               Yes.  inf?
2092 12AF3 60            GOYES   RJUS05          Yes
2093 12AF5 6052          GOTO    TIMERR          No.  NAN.
2094 12AF9 AF0   RJUS05  A=0     W
2095 12AFC A7C           A=A-1   W               Return nines
2096 12AFF 02            RTNSC                   And carry set
2097 12B01 3251  RJUS07  LCHEX   515
           5
2098 12B06 9AE           ?A>=C   XS              Exponent negative?
2099 12B09 43            GOYES   RJUSZR          Yes.  return zero.
2100 12B0B AA2           C=0     XS
2101 12B0E B32           C=C-A   X               15-exp
2102 12B11 403           GOC     RJUS99          Go if overflow
2103 12B14 AB0           A=0     X               Clear exponent field
2104 12B17 BF0           ASL     W               Position mantissa for loop
2105 12B1A A3E           C=C-1   X               Mantissa already justified?
2106 12B1D 4D1           GOC     RJUS30          Yes.
2107 12B20 814   RJUS10  ASRC                    Loop to right-justify mantissa
2108 12B23 A3E           C=C-1   X               Right-justified?
2109 12B26 480           GOC     RJUS20          Yes
2110 12B29 AC0           A=0     S               No.  Get rid of old LSN.
2111 12B2C 53F           GONC    RJUS10          B.E.T.
2112 12B2F A44   RJUS20  A=A+A   S               Round LSN.
2113 12B32 AC0           A=0     S               Get rid of it.
2114 12B35 500           RTNNC                   Return if no round up.
2115 12B38 B74           A=A+1   W               Round up A.
2116 12B3B 03    RJUS30  RTNCC
2117                *
2118 12B3D AF0   RJUSZR  A=0     W
2119 12B40 03            RTNCC
2120                *
2121 12B42 AF0   RJUS99  A=0     W
2122 12B45 A7C           A=A-1   W               Return all 9's
2123 12B48 03            RTNCC
2124                ********************************************************************
2125                ********************************************************************
2126                **
2127                ** Name:    TRUNCC  -  Truncate 15-digit Result
```

```
2128                **
2129                ** Category:   MTHUTL
2130                **
2131                ** Purpose:
2132                **      Truncate a 15-digit number into a 12-digit number.
2133                **      No rounding.
2134                **
2135                ** Entry:
2136                **      15-form in A/B.
2137                **      DEC mode.
2138                **
2139                ** Exit:
2140                **      C = 12-form result.
2141                **        = 0 if underflow.
2142                **        = INF if overflow.
2143                **      DEC mode.
2144                **      P=0.
2145                **      Carry clear.
2146                **
2147                ** Calls:      None.
2148                **
2149                ** Uses.......
2150                **              A,C,P.
2151                **
2152                ** Stk lvls:   0
2153                **
2154                ** Algorithm:
2155                **      If abs(A[A]) <= 499 then C=A W; C=B M; RTNCC
2156                **      If 499 < A[A] < 49999 then C=x000000000000F00
2157                **        (where x=A[S]); RTNCC
2158                **      C=0 W; RTNCC
2159                **
2160                ** History:
2161                **
2162                **     Date     Programmer            Modification
2163                **    --------   ----------   ---------------------------------
2164                **   05/26/82   NM           Did
2165                **
2166                ********************************************************************
2167                ********************************************************************
2168 12B4A 20    =TRUNCC P=     0
2169 12B4C 3499         LCHEX  00499
           400
2170 12B53 8BA          ?A<=C  A              0<=exponent<=499?
2171 12B56 B1           GOYES  TRUN20         Yes
2172 12B58 FA           C=-C   A
2173 12B5A 8BE          ?A>=C  A              0>exponent>=-499?
2174 12B5D 41           GOYES  TRUN20         Yes
2175 12B5F AF2          C=0    W
2176 12B62 C4           A=A+A  A
2177 12B64 4A0          GOC    TRUN10         Res=0 if exponent negative
2178 12B67 3200         LCHEX  F00            Res=inf if exponent positive
           F
2179 12B6C AC6          C=A    S              Copy sign
2180 12B6F 03     TRUN10 RTNCC
```

```
2181 12B71 AF6   TRUN20 C=A    W
2182 12B74 AD9          C=B    M
2183 12B77 03           RTNCC
2184               ****************************************************************
2185               ****************************************************************
2186               **
2187               ** Name:    ACLC24  -  A=C; C,D=24 Hrs In 512ths
2188               **
2189               ** Category:   LOCAL
2190               **
2191               ** Purpose:
2192               **      Copy C to A.  Set C,D = constant for 24 hours in
2193               **      512ths.
2194               **
2195               ** Entry:
2196               **      None.
2197               **
2198               ** Exit:
2199               **      A=C on entry.
2200               **      C,D=2A30000.
2201               **      P=4.
2202               **      Carry unaffected.
2203               **
2204               ** Calls:      None.
2205               **
2206               ** Uses.......
2207               **              A,C,D,P.
2208               **
2209               ** Stk lvls:   0
2210               **
2211               ** History:
2212               **
2213               **    Date      Programmer              Modification
2214               **  --------    ----------   ------------------------------------
2215               **  10/04/82    NM           Wrote.
2216               **
2217               ****************************************************************
2218               ****************************************************************
2219 12B79 AFA    ACLC24 A=C    W
2220 12B7C AF2           C=0    W
2221 12B7F 24            P=     4
2222 12B81 323A          LCHEX  2A3
           2
2223 12B86 AF7           D=C    W
2224 12B89 01            RTN
```

```
                        STITLE BASIC routines
                ****************************************************************
                ****************************************************************
                **
                ** Name:    TIME    -  TIME Function Execute.
                **
                ** Category:   FNEXEC
                **
                ** Purpose:
                **      Execute BASIC TIME function.
                **
                ** Entry:
                **      D0 = PC (BASIC program counter).
                **      D1 = Mathstack pointer.
                **      Jumped on TIME token.
                **
                ** Exit:
                **      Through FNRTN1.
                **
                ** Calls:      STDOD1, CMPT, ACLC24, IDIV, HEXDEC, FLOATA,
                **             DV2-12, CLRFRC, TRUNCC, RCDOD1.
                **
                ** Stk Lvls:   3
                **
                ** Detail:
                **      Result is floating-point #secs since midnight rounded
                **      to hundredths of a second.
                **
                ** Algorithm:
                **      Read time (CMPT).
                **      Extract time-of-day in 512ths.
                **      Make result floating-point decimal.
                **      Divide by 5.12.
                **      Truncate to integer.
                **      Divide by 100.0.
                **      FNRTN1 exit.
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ----------------------------------
                **  06/11/82   NM           Added documentation
                **
                ****************************************************************
                ****************************************************************
2270 12B8B 00           NIBHEX 00             No parameters
2271 12B8D 7BCE =TIME   GOSUB  STDOD1         Stash data pointers
2272 12B91 7D1A         GOSUB  CMPT           Time in 1/512 secs
2273 12B95 70EF         GOSUB  ACLC24         A=C; C,D=24 hours
2274 12B99 7D31         GOSUB  idiv
2275 12B9D 7B41         GOSUB  HEXDEC         To decimal
2276 12BA1 AFA          A=C    W
2277 12BA4 7DAE         GOSUB  FLOATA         Make a floating point
2278 12BA8 AF2          C=0    W
2279 12BAB 2C           P=     12
```

```
2280 12BAD 3221          LCHEX  512           C=5.12
           5
2281 12BB2 8E00          GOSUBL =DV2-12       Time*100
           00
2282 12BB8 8E00          GOSUBL =CLRFRC       Truncate
           00
2283 12BBE 788F          GOSUB  TRUNCC        Cut to 12-digit form
2284 12BC2 A3E           C=C-1  X
2285 12BC5 A3E           C=C-1  X             Time
2286 12BC8 77BE  TIME10  GOSUB  RCDOD1
2287 12BCC 8C00          GOLONG =FNRTN1
           00
2288           ****************************************************************
2289           ****************************************************************
2290           **
2291           ** Name:    DATE    -  DATE Function Execute.
2292           **
2293           ** Category:   FNEXEC
2294           **
2295           ** Purpose:
2296           **      Execute BASIC DATE function.
2297           **
2298           ** Entry:
2299           **      D0 = PC.
2300           **      D1 = Mathstack pointer.
2301           **      Jumped on DATE token.
2302           **
2303           ** Exit:
2304           **      Through FNRTN1.
2305           **
2306           ** Calls:      STDOD1, CMPTIM, TODT, DAY2JD, FLOATA, RCDOD1.
2307           **
2308           ** Stk Lvls:   3
2309           **
2310           ** Detail:
2311           **      Result is floating-point integer of the form YYDDD,
2312           **      where YY = lower 2 digits of year and
2313           **            DDD = day# in year (1 - 365 or 366).
2314           **
2315           ** Algorithm:
2316           **      Get time (CMPT).
2317           **      Extract day# (TODT).
2318           **      Convert day# to Julian date (DAY2JD).
2319           **      Build BCD string of YYDDD (YY is 2 lower digits of yr).
2320           **      Float string (FLOATA).
2321           **      Exit through FNRTN1.
2322           **
2323           ** History:
2324           **
2325           **    Date     Programmer             Modification
2326           **  --------   ----------   ---------------------------------
2327           **  06/11/82   NM           Added documentation
2328           **
2329           ****************************************************************
2330           ****************************************************************
```

```
2331 12BD2 00           NIBHEX 00            No parameters
2332 12BD4 748E =DATE   GOSUB  STDOD1
2333 12BD8 713A         GOSUB  CMPTIM
2334 12BDC 7946         GOSUB  TODT
2335 12BE0 AF6          C=A    W
2336 12BE3 8EE1         GOSUBL DAY2JD        Get julian date
          80
2337 12BE9 F0           ASL    A
2338 12BEB F0           ASL    A
2339 12BED F0           ASL    A             Shift year
2340 12BEF ABA          A=C    X             Day-of-year
2341 12BF2 7F5E         GOSUB  FLOATA        Convert to fp
2342 12BF6 AF6          C=A    W
2343 12BF9 6ECF         GOTO   TIME10
2344            ****************************************************************
2345            ****************************************************************
2346            **
2347            ** Name:    TIME$   -  TIME$ Function Execute.
2348            **
2349            ** Category:   FNEXEC
2350            **
2351            ** Purpose:
2352            **      Execute BASIC TIME$ function.
2353            **
2354            ** Entry:
2355            **      D0 = PC.
2356            **      D1 = Mathstack pointer.
2357            **      Jumped on TIME$ token.
2358            **
2359            ** Exit:
2360            **      Through EXPR.
2361            **
2362            ** Calls:      STDOD1, CMPTIM, TODT, SECHMS, COLNIZ, RCDOD1,
2363            **             STRHDR.
2364            **
2365            ** Stk Lvls:   3
2366            **
2367            ** Detail:
2368            **      Result is string of format "HH:MM:SS".
2369            **
2370            ** Algorithm:
2371            **      Get time (CMPTIM).
2372            **      Extract time-of-day (TODT).
2373            **      Convert to H,M,S (SECHMS).
2374            **      Convert to ASCII and insert colons.
2375            **      Write string to math stack.
2376            **      Write 8-char string descriptor to math stack.
2377            **      Return through EXPR.
2378            **
2379            ** History:
2380            **
2381            **    Date     Programmer             Modification
2382            **  --------   ----------   ----------------------------------
2383            **  06/11/82   NM           Added documentation
2384            **
```

```
2385                ****************************************************************
2386                ****************************************************************
2387 12BFD 00          NIBHEX 00            No parameters
2388 12BFF 795E =TIME$ GOSUB  STDOD1        Stash datapointers
2389 12C03 760A        GOSUB  CMPTIM
2390 12C07 7E16        GOSUB  TODT
2391 12C0B 7346        GOSUB  SECHMS        To hours, mins, secs.
2392 12C0F 7E40        GOSUB  COLNIZ        Colon-ize & ascii-ize
2393 12C13 7C6E  TIM$20 GOSUB  RCDOD1       Restore datapointers
2394 12C17 AF5         B=C    W             Hold result
2395 12C1A D2          C=0    A
2396 12C1C 20          P=     0
2397 12C1E 3101        LCHEX  10            C=00010
2398 12C22 8E00        GOSUBL =STRHDR       Room for this result?
           00
2399 12C28 AF4         A=B    W             Yes.  Fetch result.
2400 12C2B 1517        DAT1=A W             Write out string.
2401 12C2F 1CF         D1=D1- 16            Point back to top of stack.
2402 12C32 8C00        GOLONG =EXPR
           00
2403                ****************************************************************
2404                ****************************************************************
2405                **
2406                ** Name:    DATE$   -  DATE$ Function Execute.
2407                **
2408                ** Category:   FNEXEC
2409                **
2410                ** Purpose:
2411                **      Execute BASIC DATE$ function.
2412                **
2413                ** Entry:
2414                **      D0 = PC.
2415                **      D1 = Mathstack pointer.
2416                **      Jumped on DATE$ token.
2417                **
2418                ** Exit:
2419                **      Through EXPR.
2420                **
2421                ** Calls:      STDOD1, CMPTIM, TODT, DAYYMD, COLNIZ
2422                **
2423                ** Stk Lvls:   3
2424                **
2425                ** Detail:
2426                **      Result is string of form "YY/MM/DD".
2427                **
2428                ** Algorithm:
2429                **      Get time (CMPTIM).
2430                **      Extract day# (TODT).
2431                **      Convert to Y,M,D (DAYYMD).
2432                **      Turn into ASCII, insert colons (COLNIZ).
2433                **      Convert colons into slashes.
2434                **      Put result and string descriptor on stack.
2435                **      Exit through EXPR.
2436                **
2437                ** History:
```

```
2438                **
2439                **    Date     Programmer          Modification
2440                ** --------   ----------  ---------------------------------
2441                ** 06/11/82   NM          Added documentation
2442                **
2443                ********************************************************************
2444                ********************************************************************
2445 12C38 00              NIBHEX 00
2446 12C3A 7E1E =DATE$     GOSUB  STDOD1        Stash datapointers
2447 12C3E 7BC9            GOSUB  CMPTIM
2448 12C42 73E5            GOSUB  TODT
2449 12C46 AF6             C=A    W             Extract day#
2450 12C49 78E6            GOSUB  DAYYMD        To year, month, day
2451 12C4D 7010            GOSUB  COLNIZ        Colon-ize and ascii-ize
2452 12C51 2A              P=     10
2453 12C53 31F2            LCHEX  2F            Turn ":" into "/"
2454 12C57 24              P=     4
2455 12C59 31F2            LCHEX  2F            Ditto
2456 12C5D 65BF            GOTO   TIM$20
2457                ********************************************************************
2458                ********************************************************************
2459                **
2460                ** Name:    COLNIZ  -  Colon-ize And ASCII-ize BCD String
2461                **
2462                ** Category:   LOCAL
2463                **
2464                ** Purpose:
2465                **      Create output strings for TIME$, DATE$.  Given BCD
2466                **      digits, converts them to ASCII and inserts ":"s.
2467                **
2468                ** Entry:
2469                **      A[B]=AA. (BCD digits)
2470                **      B[B]=BB.
2471                **      D[B]=DD.
2472                **
2473                ** Exit:
2474                **      C = "AA:BB:DD".
2475                **      P=0.
2476                **      Carry set.
2477                **
2478                ** Calls:      YMDH10.
2479                **
2480                ** Uses.......
2481                **              C,P.
2482                **
2483                ** Stk lvls:   1
2484                **
2485                ** Algorithm:
2486                **      Arrange digits in proper order, with A representing
2487                **        ":".
2488                **      Space out digits to every other digit (C[0],[2], ...
2489                **        [14]).
2490                **      Insert 3's in odd-numbered digits (turns digits into
2491                **        ASCII, turns "A"'s into ":"'s.
2492                **
```

```
2493                ** History:
2494                **
2495                **    Date     Programmer          Modification
2496                ** --------   ----------  ----------------------------------
2497                ** 06/11/82   NM          Added documentation
2498                **
2499                ********************************************************************
2500                ********************************************************************
2501 12C61 75A4  COLNIZ GOSUB  YMDH10         C=AABBDD
2502 12C65 BF2          CSL    W
2503 12C68 BF2          CSL    W              C=AABBDD00
2504 12C6B 25           P=     5
2505 12C6D B96          CSR    WP
2506 12C70 30A          LCHEX  A              C=AA:BBDD0
2507 12C73 22           P=     2
2508 12C75 B96          CSR    WP
2509 12C78 30A          LCHEX  A              C=AA:BB:DD
2510 12C7B 28           P=     8              Prepare to expand into string
2511 12C7D 303   COLN10 LCHEX  3              Turn number in to character
2512 12C80 0D           P=P-1
2513 12C82 890          ?P=    0              Done?
2514 12C85 00           RTNYES                Yes
2515 12C87 BF2          CSL    W              No, make room for next iteration
2516 12C8A B96          CSR    WP
2517 12C8D 5FE          GONC   COLN10         B.E.T.
2518                ********************************************************************
2519                ********************************************************************
2520                **
2521                ** Name:    SETTIM  -  SETTIME Statement Execute.
2522                **
2523                ** Category:   STEXEC
2524                **
2525                ** Purpose:
2526                **      Execute BASIC SETTIME command.
2527                **
2528                ** Entry:
2529                **      D0 = PC.
2530                **      Jumped on SETTIME token.
2531                **
2532                ** Exit:
2533                **      Through NXTSTM.
2534                **      Error exits:
2535                **      =eIVARG if parameter >= 24 hours or < 0.
2536                **      =eIVARG if string parameter illegal format or numeric
2537                **        parameter complex.
2538                **
2539                ** Detail:
2540                **      SETTIME <numeric parm>|<string parm>
2541                **        <numeric parm> = secs since midnight
2542                **        <string parm> = "HH:MM:SS".
2543                **
2544                ** Algorithm:
2545                **      EXPEXC.
2546                **      Extract time-of-day from input (DECTIM).
2547                **      Check range.
```

```
2548                **      Get day#.
2549                **      Create time from day# and time-of-day.
2550                **      Set time.
2551                **      Exit.
2552                **
2553                ** History:
2554                **
2555                **    Date     Programmer          Modification
2556                **  --------   ----------   ----------------------------------
2557                **  06/15/82   NM           Added documentation
2558                **
2559                ********************************************************************
2560                ********************************************************************
2561 12C90 0000           REL(5) =FIXDC
            0
2562 12C95 0000           REL(5) =CLKPRS
            0
2563 12C9A 7A50 =SETTIM   GOSUB  expexc
2564 12C9E 72C1           GOSUB  DECTIM          Get time from input
2565 12CA2 4D6            GOC    ADJAA1          Go if error
2566 12CA5 97A            ?C=0   W               Arg = 0?
2567 12CA8 70             GOYES  SETI10          Yes.  Don't check for negative.
2568 12CAA 871            ?ST=1  1               Negative?
2569 12CAD A6             GOYES  ADJAA2          Yes.  Out of range.
2570 12CAF 76CE  SETI10   GOSUB  ACLC24          A=C; C,D=24 hours
2571 12CB3 9FE            ?A>=C  W               Time>=24 hours?
2572 12CB6 16             GOYES  ADJAA2          Yes.  error.
2573 12CB8 102            R2=A                   Save new time-of-day
2574 12CBB 111            A=R1                   Get current time
2575 12CBE 7810           GOSUB  idiv            Day# in a
2576 12CC2 AFB            C=D    W
2577 12CC5 7710           GOSUB  mpy             Midnight of day# in c
2578 12CC9 112            A=R2                   Get new time-of-day
2579 12CCC A72            C=A+C  W               New time
2580 12CCF 10A            R2=C
2581 12CD2 7F4B           GOSUB  SETIME          Set new time
2582 12CD6 6FE0           GOTO   STMEND
2583                *
2584 12CDA 8C00  idiv     GOLONG =IDIV
            00
2585 12CE0 8C00  mpy      GOLONG =MPY
            00
2586 12CE6 8C00  DECHEX   GOLONG =DCHXW
            00
2587 12CEC 8C00  HEXDEC   GOLONG =HXDCW
            00
2588 12CF2 8C00  argsta   GOLONG =ARGSTA
            00
2589 12CF8 8C00 =expexc   GOLONG =EXPEXC
            00
2590                *
2591                ********************************************************************
2592                ********************************************************************
2593                **
2594                ** Name:    ADJAAA  -  ADJABS Statement Execute.
```

```
2595                    **
2596                    ** Category:   STEXEC
2597                    **
2598                    ** Purpose:
2599                    **      Execute BASIC ADJABS command.  Performs an absolute
2600                    **      adjust (no accuracy correction) on system clock.
2601                    **
2602                    ** Entry:
2603                    **      DO = PC.
2604                    **      Jumped on ADJABS token.
2605                    **
2606                    ** Exit:
2607                    **      Through NXTSTM.
2608                    **      =eIVARG if complex numeric parm or bad format string
2609                    **        parm.
2610                    **      =eIVARG if abs(arg) >= 100 hours.
2611                    **
2612                    ** Detail:
2613                    **      ADJABS <numeric parameter>|<string parameter>
2614                    **        <numeric parameter> is # seconds to adjust (+ or -)
2615                    **        <string parm> is of form "HH:MM:SS" or "-HH:MM:SS".
2616                    **
2617                    ** Algorithm:
2618                    **      EXPEXC.
2619                    **      Decode time from input.
2620                    **      Error if correction >= 100 hours + or -.
2621                    **      Add or subtract to/from current time based on sign.
2622                    **      Perform adjust (ADJA).
2623                    **      Exit.
2624                    **
2625                    ** History:
2626                    **
2627                    **    Date     Programmer              Modification
2628                    **   --------   ----------   ------------------------------------
2629                    **   06/15/82   NM           Added documentation
2630                    **
2631                    ************************************************************
2632                    ************************************************************
2633 12CFE 0000                REL(5) =FIXDC
           0
2634 12D03 0000                REL(5) =CLKPRS
           0
2635 12D08 7CEF =ADJAAA GOSUB  expexc
2636 12D0C 7451         GOSUB  DECTIM          Decode time from input
2637 12D10 453   ADJAA1 GOC    TIMERR
2638 12D13 7730         GOSUB  ADSUBT          Add or subtract correction
2639 12D17 4E2   ADJAA2 GOC    TIMERR
2640 12D1A 7C7B         GOSUB  ADJA            Apply
2641 12D1E 67A0         GOTO   STMEND
2642                    ************************************************************
2643                    ************************************************************
2644                    **
2645                    ** Name:    ADJNNN  -  ADJUST Statement Execute.
2646                    **
2647                    ** Category:   STEXEC
```

```
2648                    **
2649                    ** Purpose:
2650                    **      Execute BASIC ADJUST command.
2651                    **
2652                    ** Entry:
2653                    **      D0 = PC.
2654                    **      Jumped on ADJUST token.
2655                    **
2656                    ** Exit:
2657                    **      Through NXTSTM.
2658                    **      Error exits:
2659                    **      =eIVARG if complex numeric parm or bad format string
2660                    **        parm.
2661                    **      =eIVARG if abs(arg) >= 100 hours.
2662                    **
2663                    ** Detail:
2664                    **      ADJUST <numeric parameter>|<string parameter>
2665                    **        <numeric parameter> is # seconds to adjust (+ or -)
2666                    **        <string parm> is of form "HH:MM:SS" or "-HH:MM:SS".
2667                    **      Performs a normal adjust (with accuracy correction)
2668                    **        on system clock.
2669                    **
2670                    ** Algorithm:
2671                    **      EXPEXC.
2672                    **      Decode time from input.
2673                    **      Add or subtract to/from current time based on sign.
2674                    **      Error if correction >= 100 hours + or -.
2675                    **      Perform adjust (ADJN).
2676                    **      Exit.
2677                    **
2678                    ** History:
2679                    **
2680                    **    Date     Programmer              Modification
2681                    **  --------   ----------   ----------------------------------
2682                    **  06/15/82   NM           Added documentation
2683                    **
2684                    ********************************************************************
2685                    ********************************************************************
2686 12D22 0000                REL(5) =FIXDC
           0
2687 12D27 0000                REL(5) =CLKPRS
           0
2688 12D2C 78CF =ADJNNN GOSUB  expexc
2689 12D30 7031         GOSUB  DECTIM
2690 12D34 411          GOC    TIMERR
2691 12D37 7310         GOSUB  ADSUBT
2692 12D3B 4A0          GOC    TIMERR
2693 12D3E 73EA         GOSUB  ADJN
2694 12D42 6380         GOTO   STMEND
2695                    ********************************************************************
2696                    ** TIMERR: Error exit with =eIVARG.
2697                    ********************************************************************
2698 12D46 20    TIMERR P=     0
2699 12D48 8C00 =Invarg GOLONG =ARGERR
           00
```

```
2700                ****************************************************************
2701                ****************************************************************
2702                **
2703                ** Name:    ADSUBT  -  Add Or Subtract Time Correction
2704                **
2705                ** Category:   LOCAL
2706                **
2707                ** Purpose:
2708                **      Add or subtract correction to current time based on
2709                **      +/- flag.  Used to apply correction being requested
2710                **      in ADJUST and ADJABS statements.
2711                **
2712                ** Entry:
2713                **      Output of DECTIM (assuming no error):
2714                **        C = ABS(Correction amount) (HEX ticks).
2715                **        R1 = Current time.
2716                **        R0 =Timer value in R0 (not used here).
2717                **        S0 = 0 for positive correction,
2718                **             1 for negative correction.
2719                **
2720                ** Exit:
2721                **      Carry set: correction >= 100 hours (failure).
2722                **      Carry clear: R2 = corrected time.
2723                **        (ready to call ADJA or ADJN).
2724                **
2725                ** Calls:      LC9999.
2726                **
2727                ** Uses.......
2728                **             A,C,P,R2.
2729                **
2730                ** Stk lvls:   1
2731                **
2732                ** Algorithm:
2733                **      Compare to 100 hours.  RTNC if too big.
2734                **      If S0 = 1 then add to current time; else subtract from
2735                **        current time.
2736                **      Result to R2.
2737                **      Return.
2738                **
2739                ** History:
2740                **
2741                **    Date     Programmer            Modification
2742                **  --------   ----------   ---------------------------------
2743                **  06/15/82   NM           Added documentation
2744                **
2745                ****************************************************************
2746                ****************************************************************
2747 12D4E AFA    ADSUBT A=C     W
2748 12D51 AF2           C=0     W
2749 12D54 23            P=      3
2750 12D56 338C          LCHEX   AFC8           100 hours (halfway to an...
           FA
2751 12D5C 9FE           ?A>=C   W                        instrument rating)
2752 12D5F 00            RTNYES                 Go if arg too big
2753 12D61 119           C=R1                   Current time
```

```
2754 12D64 A7A          A=C+A  W            Add correction
2755 12D67 861          ?ST=0  1            Positive correction?
2756 12D6A E0           GOYES  ADSUB1       Yes
2757 12D6C BF8          A=-A   W            A=-(correction+time)
2758 12D6F A7A          A=A+C  W            A=-correction
2759 12D72 A7A          A=A+C  W            A=time-correction
2760 12D75 471          GOC    ADSUB3       Corrected time is valid (>=0)
2761 12D78 7A8A ADSUB1  GOSUB  LC9999       End-of-time
2762 12D7C 5D0          GONC   ADSUB2       Go if negative correction
2763 12D7F 9F2          ?A<C   W            Time < end-of-time?
2764 12D82 B0           GOYES  ADSUB3       Yes.  Corrected time is valid.
2765 12D84 B7A          A=A-C  W            No.  Wrap around end-of-time
2766 12D87 B7A          A=A-C  W            Compensate for following add
2767 12D8A A7A  ADSUB2  A=A+C  W            Wrap neg correction around zero
2768 12D8D 102  ADSUB3  R2=A
2769 12D90 03           RTNCC
2770            ********************************************************************
2771            ********************************************************************
2772            **
2773            ** Name:    SETDAT  -  SETDATE Statement Execute
2774            **
2775            ** Category:   STEXEC
2776            **
2777            ** Purpose:
2778            **      Execute BASIC SETDATE command.
2779            **
2780            ** Entry:
2781            **      DO = PC.
2782            **      Jumped on SETDATE token.
2783            **
2784            ** Exit:
2785            **      Through NXTSTM.
2786            **      Error exit:
2787            **      =eIVARG if illegal date, complex arg, or bad format
2788            **        string arg.
2789            **
2790            ** Detail:
2791            **      SETDATE <numeric parm>|<string parm> .
2792            **        <numeric parm> is Julian date of format:
2793            **          YYDDD or YYYYDDD.
2794            **        <string parm> is date of format:
2795            **          "YY/MM/DD" or "YYYY/MM/DD".
2796            **
2797            ** Algorithm:
2798            **      EXPEXC.
2799            **      Decode date string from input (DECDAT).
2800            **      Convert date to time-at-midnight-of-date (multiply by
2801            **        24 hours worth of ticks).
2802            **      Extract time-of-day from current time.
2803            **      Combine time-of-day with time-at-midnight-of-date for
2804            **        new time.
2805            **      Perform absolute adjust.
2806            **      Exit.
2807            **
2808            ** History:
```

```
2809                **
2810                **    Date     Programmer              Modification
2811                **  --------   ----------   ----------------------------------
2812                **  06/15/82   NM           Added documentation
2813                **
2814                ********************************************************************
2815                ********************************************************************
2816 12D92 0000          REL(5) =FIXDC
          0
2817 12D97 0000          REL(5) =CLKPRS
          0
2818 12D9C 785F =SETDAT  GOSUB  expexc
2819 12DA0 7222          GOSUB  DECDAT          Get day# from input
2820 12DA4 41A           GOC    TIMERR
2821 12DA7 7ECD          GOSUB  ACLC24          A=C; C,D=24 hours
2822 12DAB 713F          GOSUB  mpy             Midnight of new date
2823 12DAF AFF           CDEX   W               Swap with 24 hr constant
2824 12DB2 111           A=R1                   Current time
2825 12DB5 712F          GOSUB  idiv            Extract time-of-day
2826 12DB9 AFB           C=D    W
2827 12DBC A79           C=B+C  W               Add to date
2828 12DBF 10A           R2=C
2829 12DC2 74DA          GOSUB  ADJA            Perform abs adjust
2830 12DC6 8C00 =STMEND  GOLONG =NEXTst         To NXTSTM
          00
2831                ********************************************************************
2832                ********************************************************************
2833                **
2834                ** Name:    EXACTT  -  EXACT Statement Execute
2835                **
2836                ** Category:   STEXEC
2837                **
2838                ** Purpose:
2839                **      Execute BASIC EXACT command.
2840                **
2841                ** Entry:
2842                **      D0 = PC.
2843                **      Jumped on EXACT token.
2844                **
2845                ** Exit:
2846                **      Through NXTSTM.
2847                **      Error exit:
2848                **      =eAF if EXACT results in illegal accuracy factor.
2849                **
2850                ** Detail:
2851                **      EXACT.
2852                **
2853                **      Begins a new adjustment period.  If exact has been
2854                **      executed before, attempts to compute an adjustment
2855                **      factor based on corrections applied since last exact.
2856                **      If adjustment factor is legal (>10 or <-10), stores in
2857                **      proper place in RAM, else gives error message and does
2858                **      not change AF.
2859                **
2860                ** Algorithm:
```

```
2861                **      Call EXACT.
2862                **      Exit.
2863                **
2864                ** History:
2865                **
2866                **    Date     Programmer          Modification
2867                **  --------   ----------   ----------------------------------
2868                **  06/15/82   NM           Added documentation
2869                **
2870                ********************************************************************
2871                ********************************************************************
2872 12DCC 0000           REL(5) =STOPDC
           0
2873 12DD1 0000           REL(5) =RTNCC
           0
2874 12DD6 76DA =EXACTT GOSUB  EXACT
2875 12DDA 5BE          GONC   STMEND         Go if successful
2876 12DDD 20    BADAF  P=     0
2877 12DDF 3100         LC(2)  =eAF
2878 12DE3 8C00         GOLONG =MfErr
           00
2879                ********************************************************************
2880                ********************************************************************
2881                **
2882                ** Name:    AF      -  Return AF To Result Stack
2883                **
2884                ** Category:   FNEXEC
2885                **
2886                ** Purpose:
2887                **      Execute BASIC AF function.
2888                **
2889                ** Entry:
2890                **      D0 = PC.
2891                **      D1 = Mathstack pointer.
2892                **      Jumped on AF token.
2893                **
2894                ** Exit:
2895                **      Through FNRTN1.
2896                **      Error exit:
2897                **      eIVARG if request illegal AF.
2898                **
2899                ** Calls:      CHKAF, DECHEX, FLOATA, GETAF, HEXDEC, PUTAF,
2900                **             PUTLAF, PUTLST, PUTOFO, RJUST, SOCNEG, STDOD1,
2901                **             argsta, cmpt.
2902                **
2903                ** Detail:
2904                **      AF[(new AF)]
2905                **      Stores new AF if parameter supplied.
2906                **
2907                ** History:
2908                **
2909                **    Date     Programmer          Modification
2910                **  --------   ----------   ----------------------------------
2911                **  06/11/82   NM           Added documentation
2912                **  12/16/82   NM           Incorporated SETAF into AF
```

```
2913                    **
2914                    ********************************************************************
2915                    ********************************************************************
2916 12DE9 801           NIBHEX 801            0 or 1 parameters
2917 12DEC 94A  =AF      ?C=0   S              Parameter supplied?
2918 12DEF F4            GOYES  AF05           No.  Return current AF.
2919 12DF1 7DFE          GOSUB  argsta         Pop new AF
2920 12DF5 17F           D1=D1+ 16             Inc stk ptr past it
2921 12DF8 840           ST=0   0              Positive
2922 12DFB 948           ?A=0   S              Positive?
2923 12DFE 50            GOYES  SETF10         No
2924 12E00 850           ST=1   0              Negative
2925 12E03 7BDC  SETF10  GOSUB  RJUST          Denormalize
2926 12E07 AF6           C=A    W
2927 12E0A 78DE          GOSUB  DECHEX         To HEX
2928 12E0E 7978          GOSUB  CHKAF          Check AF
2929 12E12 5AC           GONC   BADAF          Go if illegal
2930 12E15 102           R2=A                  Save new AF
2931 12E18 704C          GOSUB  STDOD1         Stash datapointers
2932 12E1C 7233          GOSUB  cmpt           Compute currtime with old AF
2933 12E20 76FB          GOSUB  PUTLAF         Last AF adjustment time=currtime
2934 12E24 7A0C          GOSUB  PUTLST         Time of last exact=currtime
2935 12E28 7A7B          GOSUB  PUTOFO         Clear error accumulator
2936 12E2C 7A9B          GOSUB  GETAF          Get old AF & set cry if neg
2937 12E30 12A           CR2EX                 Retrieve new AF
2938 12E33 7CBB          GOSUB  PUTAF          Store new AF
2939 12E37 11A           C=R2                  Get old AF
2940 12E3A 6B00          GOTO   AF07           CRY still intact from GETAF
2941 12E3E 7A1C  AF05    GOSUB  STDOD1         Stash datapointers
2942 12E42 748B          GOSUB  GETAF
2943 12E46 7DC9  AF07    GOSUB  SOCNEG         Set SO & C by CRY
2944 12E4A 7E9E          GOSUB  HEXDEC         Decimal AF
2945 12E4E AFA           A=C    W
2946 12E51 700C          GOSUB  FLOATA         FP
2947 12E55 AF6           C=A    W
2948 12E58 860           ?ST=0  0              Is AF positive?
2949 12E5B 50            GOYES  AF20           Yes
2950 12E5D A4E           C=C-1  S              No, sign = 9
2951 12E60 676D  AF20    GOTO   TIME10
2952                    ********************************************************************
2953                    ********************************************************************
2954                    **
2955                    ** Name:    DECTIM  -  Decode Time From Input
2956                    **
2957                    ** Category:   LOCAL
2958                    **
2959                    ** Purpose:
2960                    **      Decode time string passed to SETTIME, ADJUST, ADJABS.
2961                    **
2962                    ** Entry:
2963                    **      D0 = PC.
2964                    **      D1 = Mathstack pointer.
2965                    **      Input expression for command has already been evaluated
2966                    **        and is on the mathstack (string or numeric).
2967                    **
```

```
2968                ** Exit:
2969                **      Carry set: Failed to decode input.
2970                **      Carry clear: C = argument in 512ths.
2971                **      S0=0 if numeric input
2972                **         1 if string
2973                **      S1=0 if positive
2974                **         1 if negative
2975                **      If input was string (S0=1)
2976                **      S2=0 if no sign was given (implicit positive)
2977                **         1 if sign was given (positive or negative).
2978                **
2979                ** Calls:      GETDIG, HMSSEC, MP2-12, RCDOD1, RJUST, STDOD1,
2980                **              TDPARS, TRUNCC, argsta, cmpt.
2981                **
2982                ** Uses.......
2983                **              A,B,C,D,P,D0,D1,R0,R1,S0-S11.
2984                **
2985                ** Stk lvls:   2
2986                **
2987                ** Detail:
2988                **      Used to parse input string/numeric expression for
2989                **      SETTIME, ADJUST, ADJABS.
2990                **
2991                ** Algorithm:
2992                **      Stash D0 and D1; CMPT; Restore D0 and D1.
2993                **      If result is string; goto 1.
2994                **      Fetch numeric result from stack.
2995                **      Multiply by 512.0 (convert to ticks).
2996                **      S0=0 (numeric input).
2997                **      S1=0 if positive, 1 if negative.
2998                **      Unfloat abs(result); convert to HEX.
2999                **      RTNCC.
3000                **    1: C[B] = ":".
3001                **      Extract data from input string (TDPARS).
3002                **      Extract 3 digit strings from result (GETDIG).
3003                **      Convert H, M, S to seconds (HMSSEC).
3004                **      Convert to ticks.
3005                **      RTNCC.
3006                **
3007                ** History:
3008                **
3009                **    Date      Programmer              Modification
3010                **  --------   ----------   ----------------------------------
3011                **  06/15/82   NM           Added documentation
3012                **
3013                ********************************************************************
3014                ********************************************************************
3015 12E64 74FB  DECTIM GOSUB  STDOD1          Stash D0,D1
3016 12E68 76E2          GOSUB  cmpt           Compute time
3017 12E6C 731C          GOSUB  RCDOD1         Recall D0,D1
3018 12E70 1534          A=DAT1 S              Read first nib at stack
3019 12E74 B44           A=A+1  S              String?
3020 12E77 483           GOC    PRSTIM         Yes.
3021 12E7A 747E          GOSUB  argsta         No.  Pop w/exception chking.
3022 12E7E 2C            P=     12
```

```
3023 12E80 AF2          C=0    W
3024 12E83 3421         LCHEX  20512          C=512.0
           502
3025 12E8A 8E00         GOSUBL =MP2-12        Convert to 512ths
           00
3026 12E90 840          ST=0   0              Numeric input
3027 12E93 841          ST=0   1              Positive
3028 12E96 948          ?A=0   S              Result positive?
3029 12E99 50           GOYES  DECT10          Yes
3030 12E9B 851          ST=1   1              No.  mark negative
3031 12E9E 78AC  DECT10 GOSUB  TRUNCC
3032 12EA2 AFA          A=C    W
3033 12EA5 793C         GOSUB  RJUST          Unfloat result
3034 12EA9 AF6          C=A    W
3035 12EAC 693E         GOTO   DECHEX         Abs(result) in HEX
3036 12EB0 31A3  PRSTIM LCASC  \:\            Time separator
3037 12EB4 7930         GOSUB  TDPARS         Pull data from string
3038 12EB8 400          RTNC                  Return if fail
3039 12EBB 22           P=     2
3040 12EBD 7AC0         GOSUB  GETDIG         Get seconds
3041 12EC1 400          RTNC                  Return if fail
3042 12EC4 AF7          D=C    W              Hold in D
3043 12EC7 22           P=     2
3044 12EC9 7EB0         GOSUB  GETDIG         Get minutes
3045 12ECD 400          RTNC
3046 12ED0 AF5          B=C    W
3047 12ED3 22           P=     2
3048 12ED5 B04          A=A+1  P              Expect "E" here
3049 12ED8 7FA0         GOSUB  GETDIG         Get hours
3050 12EDC 400          RTNC
3051 12EDF AFA          A=C    W
3052 12EE2 7E83         GOSUB  HMSSEC         Convert to secs
3053 12EE6 BF2          CSL    W
3054 12EE9 BF2          CSL    W
3055 12EEC A76          C=C+C  W              512ths
3056 12EEF 03           RTNCC
3057            ****************************************************************
3058            ****************************************************************
3059            **
3060            ** Name:    TDPARS  -  Extract Numbers From Input String
3061            **
3062            ** Category:   LOCAL
3063            **
3064            ** Purpose:
3065            **      Extract numbers from input string passed to SETTIME,
3066            **      ADJUST, ADJABS and SETDATE.
3067            **
3068            ** Entry:
3069            **      D1 points at string descriptor in result stack (ready
3070            **        for POP1S).
3071            **      C[B] = Separator Character.
3072            **
3073            ** Exit:
3074            **      Carry set: Failure (bad format).
3075            **      Carry clear:
```

```
3076                **        A contains BCD digits with "F"s in place of
3077                **          separators and "E" preceding first digit.
3078                **          Example:
3079                **            Input: "12:34:56"     Output: 0000000E12F34F56
3080                **              (with separator = ":"),
3081                **            Input: "1982/06/18"   Output: 00000E1982F06F18
3082                **              (with separator = "/"),
3083                **            Input: "12:34.56"     RTNSC (failure)
3084                **              (with separator = ":").
3085                **
3086                ** Calls:      POP1S, DRANGE.
3087                **
3088                ** Uses.......
3089                **             A,B,C,D,D1
3090                **
3091                ** Stk lvls:   1
3092                **
3093                ** Detail:
3094                **      Routine is called from DECTIM and DECDAT if input
3095                **        is a string.
3096                **
3097                ** History:
3098                **
3099                **    Date     Programmer            Modification
3100                **  --------   ----------   ----------------------------------
3101                **  06/18/82   NM           Added documentation
3102                **
3103                ******************************************************************
3104                ******************************************************************
3105 12EF1 AE7    TDPARS D=C    B              Save separator
3106 12EF4 8E00          GOSUBL =POP1S         Pop string
           00
3107 12EFA 137           CD1EX
3108 12EFD C2            C=A+C  A
3109 12EFF 137           CD1EX                 Point at start of string
3110 12F02 20            P=     0
3111 12F04 3441          LC(5)  2*10           Maximum allowable result length
           000
3112 12F0B 8B6           ?A>C   A              Result too long?
3113 12F0E 00            RTNYES                Yes.  failure.
3114 12F10 850           ST=1   0              Result is string
3115 12F13 841           ST=0   1              Result is positive
3116 12F16 842           ST=0   2              No sign given
3117 12F19 81C           ASRB                  #bytes in result
3118 12F1C D8            B=A    A              Byte counter
3119 12F1E AF0           A=0    W              Room for result
3120 12F21 A2C           A=A-1  XS
3121 12F24 A2C           A=A-1  XS             Mark start of string with "e"
3122 12F27 1C1           D1=D1- 2              Point at first char
3123 12F2A CD            B=B-1  A
3124 12F2C 400           RTNC                  Failure if null
3125 12F2F 14B           A=DAT1 B              Read in first character
3126 12F32 3102          LCASC  \ \
3127 12F36 962           ?A=C   B              First char=blank?
3128 12F39 91            GOYES  TDPA20         Yes.  positive, no sign given
```

```
3129 12F3B 31B2          LCASC  \+\
3130 12F3F 962           ?A=C   B              First char="+"?
3131 12F42 D0            GOYES  TDPA10         Yes.  positive, sign given
3132 12F44 30D           LCHEX  D
3133 12F47 966           ?A#C   B              First char="-"
3134 12F4A 31            GOYES  TDPA30         No.  must be number.
3135 12F4C 851           ST=1   1              Negative
3136 12F4F 852   TDPA10  ST=1   2              Sign given
3137 12F52 1C1   TDPA20  D1=D1- 2              Start of loop
3138 12F55 CD            B=B-1  A
3139 12F57 4B2           GOC    TDPA60         Go if done
3140 12F5A 14B           A=DAT1 B              Read next char
3141 12F5D AEB   TDPA30  C=D    B              Separator
3142 12F60 966           ?A#C   B              Is this separator?
3143 12F63 C0            GOYES  TDPA40         No
3144 12F65 AE0           A=0    B
3145 12F68 A6C           A=A-1  B              Yes.  mark with f.
3146 12F6B 6010          GOTO   TDPA50
3147 12F6F 8F00  TDPA40  GOSBVL =DRANGE        Check for character = digit.
           000
3148 12F76 400           RTNC
3149 12F79 BE0           ASL    B
3150 12F7C BF0   TDPA50  ASL    W              Shift in digit or "f"
3151 12F7F 62DF          GOTO   TDPA20
3152 12F83 BF4   TDPA60  ASR    W
3153 12F86 BF4           ASR    W              Position result
3154 12F89 03            RTNCC
3155             ********************************************************************
3156             ********************************************************************
3157             **
3158             ** Name:    GETDIG  -  Extract Digits From TDPARS Output
3159             **
3160             ** Category:   LOCAL
3161             **
3162             ** Purpose:
3163             **      Extract fields (those things separated by field
3164             **      separators) from the BCD string created by TDPARS.
3165             **
3166             ** Entry:
3167             **      A contains output from TDPARS.
3168             **      P points at separator above lowest field in A
3169             **        (field to be extracted).
3170             **      HEX mode.
3171             **
3172             ** Exit:
3173             **      Carry set: failure (P not pointing at field separator
3174             **        or there is a non-BCD digit in field).
3175             **      Carry clear: C=result (number pulled from A).
3176             **      Field has been SR'd out of A.
3177             **
3178             ** Calls:      None.
3179             **
3180             ** Uses.......
3181             **            A,C,P.
3182             **
```

```
3183                ** Stk lvls:   0
3184                **
3185                ** Detail:
3186                **      Field separator is a HEX F.
3187                **
3188                ** History:
3189                **
3190                **    Date      Programmer              Modification
3191                **  --------   ----------   ---------------------------------
3192                **  06/28/82   NM           Added documentation
3193                **
3194                ********************************************************************
3195                ********************************************************************
3196 12F8B 80FF   GETDIG CPEX   15              Save p
3197 12F8F 80DF         P=C    15
3198 12F93 B04          A=A+1  P               Pointing at separator?
3199 12F96 5D2          GONC   GETDFL          No.  error.
3200 12F99 05           SETDEC
3201 12F9B A92          C=0    WP
3202 12F9E A1E          C=C-1  WP              9's
3203 12FA1 04           SETHEX
3204 12FA3 0D     GETD10 P=P-1                  Point at next digit
3205 12FA5 4A0          GOC    GETD20
3206 12FA8 986          ?A>C   P               Legal bcd digit?
3207 12FAB 00           RTNYES                 No.  failure.
3208 12FAD 55F          GONC   GETD10          B.E.T.
3209 12FB0 80DF   GETD20 P=C    15             Retrieve pointer
3210 12FB4 AF2          C=0    W
3211 12FB7 A96          C=A    WP              Copy number
3212 12FBA BF4    GETD30 ASR    W              Shift off number
3213 12FBD 0D           P=P-1
3214 12FBF 5AF          GONC   GETD30
3215 12FC2 03           RTNCC
3216 12FC4 02     GETDFL RTNSC
3217                ********************************************************************
3218                ********************************************************************
3219                **
3220                ** Name:    DECDAT  -  Decode Date From Input Num/string
3221                **
3222                ** Category:   LOCAL
3223                **
3224                ** Purpose:
3225                **      Decode date input to SETDATE.
3226                **
3227                ** Entry:
3228                **      D0 = PC.
3229                **      D1 = Mathstack pointer.
3230                **      Expression has been executed (result on mathstack).
3231                **
3232                ** Exit:
3233                **      Time in R1.
3234                **      Timer value in R0.
3235                **      Carry set: failure (bad format or date).
3236                **      Carry clear:  C contains day# (HEX days since day 0).
3237                **
```

```
3238                ** Calls:      ASRW3, DAYYMD, GETDIG, JD2DAY, RANGYR, RCDOD1,
3239                **             RJUST, STDOD1, TDPARS, YMDDAY, YMDH20, argsta,
3240                **             cmpt.
3241                **
3242                ** Uses.......
3243                **  Exclusive: A,B,C,D,P,D0,D1,R0,R1,S0-S11.
3244                **
3245                ** Stk lvls:   2
3246                **
3247                ** Detail:
3248                **      Numeric input is interpreted as Julian date YYDDD or
3249                **        YYYYDDD.
3250                **      String input is "YY/MM/DD" or "YYYY/MM/DD".
3251                **      If year on numeric input is <100 or string input is
3252                **        given as YY (2-digits only), year is transformed
3253                **        into the range [1960 .. 2059].
3254                **
3255                ** History:
3256                **
3257                **    Date      Programmer              Modification
3258                **  --------   ----------   ---------------------------------
3259                **  06/18/82   NM           Added documentation
3260                **
3261                ********************************************************************
3262                ********************************************************************
3263 12FC6 729A  DECDAT GOSUB  STDOD1          Stash pointers
3264 12FCA 7481         GOSUB  cmpt           Compute time
3265 12FCE 71BA         GOSUB  RCDOD1         Restore pointers
3266 12FD2 1534         A=DAT1 S              Read low nibble at stack
3267 12FD6 B44          A=A+1  S              String?
3268 12FD9 4B5          GOC    PRSDAT         Yes.
3269 12FDC 721D         GOSUB  argsta         No.  Pop with exception chking.
3270 12FE0 3260         LCHEX  006            Is julian date
           0
3271 12FE5 9B6          ?A>C   X              Exponent too large?
3272 12FE8 00           RTNYES                Yes... failure
3273 12FEA 94C          ?A#0   S              Positive?
3274 12FED 00           RTNYES                No, error.
3275 12FEF 7FEA         GOSUB  RJUST          Denormalize
3276 12FF3 AF1          B=0    W
3277 12FF6 AB8          B=A    X              Copy day-in-year
3278 12FF9 8E00         GOSUBL =ASRW3
           00
3279 12FFF 7DA0         GOSUB  RANGYR         Correct year
3280 13003 96C          ?A#0   B              Year divisible by 100?
3281 13006 40           GOYES  DECD10         No.  check for div by 4.
3282 13008 22           P=     2              Yes.  check for div by 400.
3283 1300A 303   DECD10 LCHEX  3
3284 1300D 0E02         C=A&C  P              Look at lower 2 bits
3285 13011 A0E          C=C-1  P              Carry set if divisible
3286 13014 D2           C=0    A
3287 13016 20           P=     0
3288 13018 3256         LCHEX  365
           3
3289 1301D 540     .    GONC   DECD20         Go if non-leap year
```

```
3290 13020 E6          C=C+1  A        C=366.
3291 13022 9B1  DECD20 ?B>C   X        Day-of-year legal?
3292 13025 00          RTNYES          No
3293 13027 939         ?B=0   X        Maybe.  =0?
3294 1302A 00          RTNYES          Yes.  Error.
3295 1302C AB9         C=B    X        No.
3296 1302F 77C3        GOSUB  JD2DAY    Get day#
3297 13033 03          RTNCC           Done
3298 13035 31F2 PRSDAT LCASC  \/\      Date separator
3299 13039 74BE        GOSUB  TDPARS    Pull data from string
3300 1303D 400         RTNC
3301 13040 872         ?ST=1  2        Was a sign indicated?
3302 13043 00          RTNYES          Yes.  failure.
3303 13045 22          P=     2
3304 13047 704F        GOSUB  GETDIG    Get day
3305 1304B 400         RTNC
3306 1304E AF7         D=C    W
3307 13051 22          P=     2
3308 13053 743F        GOSUB  GETDIG    Get month
3309 13057 400         RTNC
3310 1305A AF5         B=C    W
3311 1305D 24          P=     4        This should be either "e" or "0"
3312 1305F 90C         ?A#0   P        0?
3313 13062 91          GOYES  PRSD10    No.  "e".
3314 13064 22          P=     2        Yes.  look for 2-digit year.
3315 13066 B04         A=A+1  P        Should become "f"
3316 13069 7E1F        GOSUB  GETDIG    Extract year
3317 1306D 400         RTNC
3318 13070 AFA         A=C    W
3319 13073 7930        GOSUB  RANGYR    Correct year
3320 13077 6010        GOTO   PRSD20
3321 1307B B04  PRSD10 A=A+1  P
3322 1307E 790F        GOSUB  GETDIG    Get year
3323 13082 400         RTNC
3324 13085 AFA         A=C    W
3325 13088 7180 PRSD20 GOSUB  YMDH20    Pack date into C.
3326 1308C 10A         R2=C            Stash
3327 1308F 7172        GOSUB  YMDDAY    Get day#
3328 13093 10B         R3=C            Save
3329 13096 7B92        GOSUB  DAYYMD    Back into date
3330 1309A 04          SETHEX
3331 1309C AF6         C=A    W
3332 1309F 7A60        GOSUB  YMDH20    Squeeze into register
3333 130A3 112         A=R2            Get original input
3334 130A6 976         ?A#C   W        Same?
3335 130A9 00          RTNYES          No.
3336 130AB 11B         C=R3            Yes.  Date is legal.
3337 130AE 03          RTNCC
3338          ****************************************************************
3339          ****************************************************************
3340          **
3341          ** Name:    RANGYR  -  Check Range Of Year
3342          **
3343          ** Category:   LOCAL
3344          **
```

```
3345                ** Purpose:
3346                **      Convert 2-digit year to a standard range.
3347                **
3348                ** Entry:
3349                **      Year in A[W] (BCD).
3350                **
3351                ** Exit:
3352                **      If year > 2 digits, year in A and C.
3353                **      Else (calling lower 2 digits YY), A contains:
3354                **          19YY if YY >= 60,
3355                **          20YY if YY < 60.
3356                **      P=0.
3357                **
3358                ** Calls:      None.
3359                **
3360                ** Uses.......
3361                **              A,C,P.
3362                **
3363                ** Stk lvls:   0
3364                **
3365                ** History:
3366                **
3367                **    Date      Programmer              Modification
3368                **  --------    ----------    ----------------------------------
3369                **  06/18/82    NM            Added documentation
3370                **
3371                ********************************************************************
3372                ********************************************************************
3373 130B0 AF2    RANGYR C=0     W
3374 130B3 AE6           C=A     B               Copy yy
3375 130B6 972           ?A=C    W               Year=00yy?
3376 130B9 70            GOYES   RANG10          Yes
3377 130BB AF6           C=A     W
3378 130BE 01            RTN
3379 130C0 20     RANG10 P=      0
3380 130C2 3306          LCHEX   1960
            91
3381 130C8 9EE           ?A>=C   B               Yy<60?
3382 130CB 80            GOYES   RANG20          No.  use 19yy.
3383 130CD 3300          LCHEX   2000            Yes.  use 20yy.
            02
3384 130D3 AE6    RANG20 C=A     B
3385 130D6 AFA           A=C     W
3386 130D9 01            RTN
```

```
3387                    $TITLE Time system utilities
3388                ********************************************************************
3389                ********************************************************************
3390                **
3391                ** Name:(S) YMDHMS  -  Return Time And Date
3392                ** Name:(S) YMDH01  -  Convert Time To YYMMDD And HHMMSS
3393                **
3394                ** Category:   TIME
3395                **
3396                ** Purpose:
3397                **   YMDHMS: Return current time and date in format compatible
3398                **           with file header time/date field.
3399                **   YMDH01: Convert passed time (seconds since year 0) into
3400                **           time/date format compatible with file header
3401                **           time/date field.
3402                **
3403                ** Entry:
3404                **   YMDHMS: None.
3405                **   YMDH01: C[W]=Time (seconds since midnight, 1 Jan 0000).
3406                **
3407                ** Exit:
3408                **       C = 0000YYMMDDHHMMSS. (year,mo,day,hrs,min,sec)
3409                **       A[B] = HH (same as HH in C).
3410                **       B[B] = MM (same as MM -- minutes in C).
3411                **       D[B] = SS (same as SS in C).
3412                **       HEX mode.
3413                **       Carry clear.
3414                **
3415                ** Calls:      CMPT, TIMRND, TODT, DAYYMD, SECHMS.
3416                **
3417                ** Uses.......
3418                **             A,B,C,D,P,D0,D1,R0,R1,S0-S11.
3419                **
3420                ** Stk lvls:   2
3421                **
3422                ** Algorithm:
3423                **       Get current time.
3424                **       Compute day#, time-of-day.
3425                **       Compute YYMMDD from day#.
3426                **       Compute HHMMSS from time-of-day.
3427                **       Format into YYMMDDHHMMSS.
3428                **
3429                ** History:
3430                **
3431                **    Date     Programmer              Modification
3432                **  --------   ----------   ------------------------------------
3433                **  60/11/82   NM           Added documentation
3434                **
3435                ********************************************************************
3436                ********************************************************************
3437 130DB 7370 =YMDHMS GOSUB  CMPT            Compute time
3438 130DF 8EC2        GOSUBL TIMRND          This saves a subroutine level
            5F
3439 130E5 7041 =YMDH01 GOSUB  TODT            To day# and time-of-day
3440 130E9 108         R0=C                   Save time-of-day
```

```
3441 130EC AF6          C=A    W
3442 130EF 7242         GOSUB  DAYYMD         Get year,month,day
3443 130F3 AF2          C=0    W
3444 130F6 7010         GOSUB  YMDH10         Format into yymmdd
3445 130FA BF2          CSL    W
3446 130FD BF2          CSL    W              YymmddOO
3447 13100 128          CROEX                 Swap with time-of-day
3448 13103 7B41         GOSUB  SECHMS
3449 13107 118          C=R0                  Result will be yymmddhhmmss
3450 1310A AE6   YMDH10 C=A    B
3451 1310D BF2   YMDH20 CSL    W
3452 13110 BF2          CSL    W
3453 13113 AE9          C=B    B
3454 13116 BF2          CSL    W
3455 13119 BF2          CSL    W
3456 1311C AEB          C=D    B
3457 1311F 03           RTNCC
3458            ****************************************************************
3459            ****************************************************************
3460            **
3461            ** Name:    INITCL  -  Initialize Clock System
3462            **
3463            ** Category:   CONFIG
3464            **
3465            ** Purpose:
3466            **      Initialize clock system at coldstart.
3467            **      Zeroes out clock system RAM, sets time to 000000000000.
3468            **
3469            ** Entry:
3470            **      None.
3471            **
3472            ** Exit:
3473            **      Through CMPT.
3474            **      Carry clear.
3475            **      P=0.
3476            **
3477            ** Calls:      CMPT (falls through).
3478            **
3479            ** Uses.......
3480            **              A,B,C,D,P,D0,D1,R0,R1,S0-S11.
3481            **
3482            ** Stk lvls:   1
3483            **
3484            ** Algorithm:
3485            **      Zero out clock system RAM.
3486            **      Set timer to 0.
3487            **      Fall through to CMPT for normal update.
3488            **
3489            ** History:
3490            **
3491            **    Date     Programmer             Modification
3492            **  --------   ----------   ----------------------------------
3493            **  06/11/82   NM           Added documentation
3494            **
3495            ****************************************************************
```

```
3496                ********************************************************************
3497 13121 1FD0    =INITCL D1=(5) =NXTIRQ
           7F2
3498 13128 20              P=     0
3499 1312A D2              C=0    A
3500 1312C 3108            LC(2)  (TIMAF)+6-(NXTIRQ)  Size of clock RAM
3501 13130 8E00            GOSUBL =WIPOUT        Clear out clock RAM
           00
3502 13136 1B8F            D0=(5) =TIMER2
           2E2
3503 1313D 27              P=     7
3504 1313F AF2             C=0    W
3505 13142 308             LCHEX  8
3506 13145 15C7            DAT0=C 8              Clear out and enable timer
3507 13149 A7E             C=C-1  W              Now write FFFFFF to timer.
3508 1314C 8E00            GOSUBL =WRTTMR        Write to timer.
           00
3509 13152 8CE5  cmpt      GOLONG CMPT           Update
           4F
3510                ********************************************************************
3511                ********************************************************************
3512                **
3513                ** Name:(S) SETTM0  -  Set System Timeout
3514                **
3515                ** Category:   TIME
3516                **
3517                ** Purpose:
3518                **      Set 10-minute system timeout.
3519                **
3520                ** Entry:
3521                **      None.
3522                **
3523                ** Exit:
3524                **      Carry set.
3525                **      HEX mode.
3526                **      10-minute timeout alarm has been scheduled.
3527                **
3528                ** Calls:      ST01, SFLAG?, SETALR, SETALM, RC01.
3529                **
3530                ** Uses.......
3531                **             A,B,C,D,P,D0,D1, SCRTCH[0-31], SCREX0.
3532                **
3533                ** Stk lvls:   3
3534                **
3535                ** Detail:
3536                **      Typically used to schedule automatic power-down.
3537                **      Also used to schedule timeout during "Align" message in
3538                **        card reader.
3539                **      If =flCTON (continuous on) flag is set, the timeout
3540                **        is disabled (never comes due).
3541                **
3542                ** Algorithm:
3543                **      Stash scratch regs.
3544                **      If flCTON set, set ALRM4 = 0 (SETALM)
3545                **        else set ALRM4 = current time + 10 minutes (SETALR).
```

```
3546                **      Clock update (CMPT).
3547                **      Restore scratch regs.
3548                **
3549                ** History:
3550                **
3551                **    Date     Programmer          Modification
3552                **  --------   ----------  ----------------------------------
3553                **  06/11/82   NM          Added documentation
3554                **
3555                ********************************************************************
3556                ********************************************************************
3557 13158 04     =SETTMO SETHEX
3558 1315A 7E49         GOSUB  ST01          Save scratch regs
3559 1315E 1B14         DO=(5) =SCREXO       Save scratch regs
           9F2
3560 13165 09           C=ST
3561 13167 144          DAT0=C A             Stash status bits
3562 1316A 8E24         GOSUBL CMPT          Compute current time
           4F
3563 13170 20           P=     0
3564 13172 3100         LC(2)  =flCTON
3565 13176 7000         GOSUB  =SFLAG?       Continuous on?
3566 1317A AF0          A=0    W
3567 1317D 421          GOC    STM100        Go if continuous on
3568 13180 111          A=R1
3569 13183 AF2          C=0    W
3570 13186 3400         LCHEX  4B000
           0B4
3571 1318D A7A          A=A+C  W             Time of alarm
3572 13190 2F    STM100 P=     15
3573 13192 303          LCHEX  3             Alarm number
3574 13195 7460         GOSUB  seta10
3575 13199 1B14         DO=(5) =SCREXO
           9F2
3576 131A0 146          C=DAT0 A
3577 131A3 0A           ST=C                 Restore status bits
3578 131A5 6029         GOTO   RC01          Restore scratch regs
3579                ********************************************************************
3580                ********************************************************************
3581                **
3582                ** Name:    ACTTMR  -  Activate On-timer
3583                **
3584                ** Category:   TIME
3585                **
3586                ** Purpose:
3587                **      Schedule an on-timer alarm.
3588                **
3589                ** Entry:
3590                **      C[0] = Timer # (timer# = 1, 2 or 3).
3591                **      A[W] = Time interval (HEX 512ths).
3592                **
3593                ** Exit:
3594                **      Through SETALR.
3595                **
3596                ** Calls:      SETALR (falls through).
```

```
3597                ;;
3598                ;; Uses.......
3599                ;;              A,B,C,D,D0,D1,R0,R1,R3,S0-S11
3600                ;;
3601                ;; Stk lvls:   2
3602                ;;
3603                ;; Detail:
3604                ;;      Schedules alarm# (on-timer# - 1) for (time interval)
3605                ;;      from current time.
3606                ;;
3607                ;; Algorithm:
3608                ;;      C=C-1 B
3609                ;;      goto SETALR
3610                ;;
3611                ;; History:
3612                ;;
3613                ;;     Date     Programmer             Modification
3614                ;;   --------   ----------   ---------------------------------
3615                ;;   05/25/82   NM           Added documentation
3616                ;;
3617                ;;*****************************************************************
3618                ;;*****************************************************************
3619 131A9 A6E   =RCTTMR C=C-1  B               Alarm# (0-2)
3620 131AC 8C96  setalr GOLONG SETALR
           7F
3621                ;;*****************************************************************
3622                ;;*****************************************************************
3623                ;;
3624                ;; Name:    RCTTM1  -  Reactivate On-timer Alarm
3625                ;;
3626                ;; Category:   TIME
3627                ;;
3628                ;; Purpose:
3629                ;;      Reschedule on-timer alarm.
3630                ;;
3631                ;; Entry:
3632                ;;      C[S] = Timer #.
3633                ;;      A[11-0] = Interval.
3634                ;;
3635                ;; Exit:
3636                ;;      Through SETALM.
3637                ;;      Carry clear.
3638                ;;      P=0.
3639                ;;      R3[S]=timer#-1.
3640                ;;
3641                ;; Calls:      CMPT, IDIV, SETALM (falls through).
3642                ;;
3643                ;; Uses.......
3644                ;;              A,B,C,D,P,D0,D1,R0,R1,R3,S0-S11.
3645                ;;
3646                ;; Stk lvls:   2
3647                ;;
3648                ;; Detail:
3649                ;;      Schedules alarm n*(interval) after last alarm time,
3650                ;;      where n is the smallest positive integer for which
```

```
3651                **      n*(interval)+(last alarm time) is after current time.
3652                **
3653                ** Algorithm:
3654                **      A = (Current time) - (Last alarm time).
3655                **      C = -(A mod (interval)) + interval  {time to next
3656                **        alarm}.
3657                **      Set alarm to (Current time) + C.
3658                **
3659                ** History:
3660                **
3661                **     Date     Programmer            Modification
3662                **   --------   ----------   ----------------------------------
3663                **   05/27/82   NM           Wrote, etc.
3664                **
3665                ************************************************************
3666                ************************************************************
3667 131B2 A4E    =RCTTM1 C=C-1  S              Convert timer# to alarm#
3668 131B5 ACA            A=C    S
3669 131B8 103            R3=A                 Stash alarm# and interval
3670 131BB 739F           GOSUB  cmpt          Compute current time
3671 131BF AFA            A=C    W             Hold time
3672 131C2 1FD0           D1=(5) (=ALRM1)-12
            7F2
3673 131C9 11B            C=R3                 Fetch interval
3674 131CC 17B   RCTT10   D1=D1+ 12            Loop to point at selected alarm
3675 131CF A4E            C=C-1  S
3676 131D2 59F            GONC   RCTT10
3677 131D5 AF5            B=C    W             Stash interval
3678 131D8 AF2            C=0    W
3679 131DB 15FB           C=DAT1 12            Read last alarm time
3680 131DF B7A            A=A-C  W             Current time - last alarm time
3681 131E2 2B             P=     11
3682 131E4 A99            C=B    WP            Lower 12 digits of interval
3683 131E7 AF7            D=C    W             Hold
3684 131EA 7CEA           GOSUB  idiv          Divide by interval
3685 131EE AFB            C=D    W
3686 131F1 B79            C=C-B  W             Time to next alarm
3687 131F4 111            A=R1                 Time
3688 131F7 A7A            A=A+C  W             Time of next alarm
3689 131FA 11B            C=R3                 Alarm# in C[S]
3690 131FD 8C03  seta10   GOLONG SETA10        Set the alarm
            7F
3691                ************************************************************
3692                ************************************************************
3693                **
3694                ** Name:    SEC2TK  -  Convert Seconds To Ticks (512ths)
3695                **
3696                ** Category:   TIME
3697                **
3698                ** Purpose:
3699                **      Convert floating-point #seconds to # 512ths (time
3700                **      system's smallest unit of measurement.
3701                **
3702                ** Entry:
3703                **      A = 12-digit floating-point # secs.
```

```
3704                **
3705                ** Exit:
3706                **      A,B,C = Hex ticks.  If # ticks >= 1E16, result is
3707                **        chopped to 1E16-1.
3708                **      HEX mode.
3709                **      P=0.
3710                **      Carry clear.
3711                **
3712                ** Calls:      MP2-12, TRUNCC, RJUST, DCHXW.
3713                **
3714                ** Uses.......
3715                **  Exclusive: A,B,C,D,P.
3716                **
3717                ** Stk lvls:   1
3718                **
3719                ** Detail:
3720                **      Because of DECmode RJUST, maximum result is HEX
3721                **      equivalent of 1E16-1.
3722                **
3723                ** Algorithm:
3724                **      Multiply arg by 512.0.
3725                **      Truncate to 12-digit form (TRUNCC).
3726                **      Unfloat (RJUST).
3727                **      Convert to HEX.
3728                **
3729                ** History:
3730                **
3731                **     Date     Programmer            Modification
3732                **   --------   ----------   ---------------------------------
3733                **   06/18/82   NM           Added documentation
3734                **
3735                ********************************************************************
3736                ********************************************************************
3737 13203 05     =SEC2TK SETDEC
3738 13205 AF2             C=0     W
3739 13208 2C              P=      12
3740 1320A 3421            LCHEX   20512          C=512.0
           502
3741 13211 8E00            GOSUBL  =MP2-12        Time in 512ths
           00
3742 13217 7F29            GOSUB   TRUNCC         To c
3743 1321B AFA             A=C     W
3744 1321E 70C8            GOSUB   RJUST          Convert to integer
3745 13222 AF6             C=A     W
3746 13225 60CA            GOTO    DECHEX         To HEX
```

```
3747                    STITLE Time and Date conversion
3748            *
3749            ******************************************************************
3750            ******************************************************************
3751            **
3752            ** Name:(S) TODT    -  Time To Time-of-day And Day#
3753            **
3754            ** Category:   TIME
3755            **
3756            ** Purpose:
3757            **      Convert from time (since 0000) to day# (since day 0)
3758            **      and time-of-day (since midnight).
3759            **
3760            ** Entry:
3761            **      C = Time (HEX seconds).
3762            **      Hex mode.
3763            **
3764            ** Exit:
3765            **      B,C = Time-of-day (HEX seconds).
3766            **      A = Day# (HEX days since day 0).
3767            **      Hex mode.
3768            **      P=15.
3769            **      Carry set.
3770            **
3771            ** Calls:      IDIV (falls through)
3772            **
3773            ** Uses.......
3774            **              A,B,C,P
3775            **
3776            ** Stk lvls:   0
3777            **
3778            ** Detail:
3779            ** The following terms are used in this and the following
3780            ** documentation:
3781            **
3782            **   time: time in seconds since midnight 1 jan 0000
3783            **   time-of-day: seconds since midnight.
3784            **   day#: day# relative to 1 jan 0000
3785            **   h,m,s: hours, minutes, seconds.
3786            **   d,m,y: day, month, year.
3787            **
3788            ** Date routines are valid from 1 jan 0000 to
3789            **      31 dec 9999.
3790            ** Assumptions being made in the date routines are:
3791            **      year<=9999
3792            **      month<=12
3793            **      day<=31 (this is intentionally violated for JD2DAY)
3794            **      day#<=3652424
3795            **      THIS MEANS THAT HIGHER-ORDER DIGITS ARE ZEROES!!
3796            **
3797            ** Algorithm:
3798            **      Day#=Time div 15180H.
3799            **      Time-of-day=Time mod 15180H.
3800            **
3801            ** History:
```

```
3802                ** 
3803                **     Date     Programmer            Modification
3804                **   --------   ----------   ---------------------------------
3805                **   05/24/82   NM           Added documentation
3806                ** 
3807                ********************************************************************
3808                ********************************************************************
3809 13229 AFA  =TODT   A=C    W               Time to C
3810 1322C AF2         C=0    W
3811 1322F 21          P=     1
3812 13231 3381        LCHEX  1518            #secs/day (divisor)
           51
3813 13237 62AA        GOTO   idiv
3814                ********************************************************************
3815                ********************************************************************
3816                ** 
3817                ** Name:    FROMDT  -  Time-of-day And Date To Time
3818                ** 
3819                ** Category:   TIME
3820                ** 
3821                ** Purpose:
3822                **      Convert from time-of-day (secs since midnight) and
3823                **      day# (days since day 0) to time (secs since time 0).
3824                ** 
3825                ** Entry:
3826                **      A = Day# (HEX days since day 0).
3827                **      C = Time-of-day (HEX seconds).
3828                **      Hex mode.
3829                ** 
3830                ** Exit:
3831                **      C = Time (HEX seconds since time 0).
3832                **      Hex mode.
3833                ** 
3834                ** Calls:      MPY.
3835                ** 
3836                ** Uses.......
3837                **             A,B,C,D,P.
3838                ** 
3839                ** Stk lvls:   1
3840                ** 
3841                ** Algorithm:
3842                **      Time=Day# * 15180H + time-of-day.
3843                ** 
3844                ** History:
3845                ** 
3846                **    Date     Programmer            Modification
3847                **  --------   ----------   ---------------------------------
3848                **  05/24/82   NM           Added documentation
3849                ** 
3850                ********************************************************************
3851                ********************************************************************
3852 1323B AF7  =FROMDT D=C    W               Copy timeofday to d
3853 1323E AF2         C=0    W
3854 13241 21          P=     1
3855 13243 3381        LCHEX  1518            Secs/day
```

```
            51
 3856 13249 739A          GOSUB  mpy            Result to c
 3857 1324D A7B           C=C+D  W
 3858 13250 01            RTN
 3859              ****************************************************************
 3860              ****************************************************************
 3861              **
 3862              ** Name:(S) SECHMS  -  Convert Secs To Hours, Mins, Secs
 3863              **
 3864              ** Category:   TIME
 3865              **
 3866              ** Purpose:
 3867              **      Convert time in seconds (expressed in HEX) to
 3868              **      hours, minutes and seconds (expressed in DEC).
 3869              **
 3870              ** Entry:
 3871              **      C[W] = Time-of-day (HEX seconds).
 3872              **
 3873              ** Exit:
 3874              **      A[W] = Hours (BCD integer).
 3875              **      B[W],C[W] = Minutes (BCD integer).
 3876              **      D[W] = Seconds (BCD integer).
 3877              **      HEX mode.
 3878              **      Carry clear.
 3879              **      P=15.
 3880              **
 3881              ** Calls:      HEXDEC, IDIV.
 3882              **
 3883              ** Uses.......
 3884              **              A,B,C,D,P.
 3885              **
 3886              ** Stk lvls:   1
 3887              **
 3888              ** Algorithm:
 3889              **      Convert to decimal.
 3890              **      Divide by 60; remainder=secs.
 3891              **      Divide quotient by 60; remainder = minutes,
 3892              **        quotient = hours.
 3893              **
 3894              ** History:
 3895              **
 3896              **     Date     Programmer             Modification
 3897              **    --------   ----------   ---------------------------------
 3898              **    05/27/82   NM           Added documentation
 3899              **
 3900              ****************************************************************
 3901              ****************************************************************
 3902 13252 769A =SECHMS  GOSUB  HEXDEC          Decimal seconds to c
 3903 13256 AFA           A=C    W
 3904 13259 7B00          GOSUB  DV60            A=minutes, b=seconds
 3905 1325D AF7           D=C    W              Remainder (secs) to d
 3906 13260 7400          GOSUB  DV60            A=hours, b=minutes
 3907 13264 04            SETHEX
 3908 13266 03            RTNCC
 3909 13268 21     DV60   P=     1
```

```
3910 1326A AF2          C=0    W
3911 1326D 306          LCHEX  6
3912 13270 696A         GOTO   idiv            Divide by 60
3913                ****************************************************************
3914                ****************************************************************
3915                **
3916                ** Name:(S) HMSSEC  -  Hours, Mins, Secs To Seconds.
3917                **
3918                ** Category:   TIME
3919                **
3920                ** Purpose:
3921                **      Convert from hours, minutes, secs (DEC) to seconds
3922                **      (HEX).
3923                **
3924                ** Entry:
3925                **      A[W] = Hours (BCD integer).
3926                **      B[W] = Minutes (BCD integer).
3927                **      D[W] = Seconds (BCD integer).
3928                **
3929                ** Exit:
3930                **      A,B,C = Seconds since midnight (HEX).
3931                **      HEX mode.
3932                **      P=0.
3933                **      Carry clear.
3934                **
3935                ** Calls:      MP60, IDIV.
3936                **
3937                ** Uses.......
3938                **              A,B,C,D,P.
3939                **
3940                ** Stk lvls:   1
3941                **
3942                ** Algorithm:
3943                **      Compute ((hrs * 60) + mins) * 60 + secs.
3944                **      Convert to HEX.
3945                **
3946                ** History:
3947                **
3948                **     Date     Programmer             Modification
3949                **   --------   ----------   ---------------------------------
3950                **   05/27/82   NM           Added documentation
3951                **
3952                ****************************************************************
3953                ****************************************************************
3954 13274 05    =HMSSEC SETDEC
3955 13276 7110         GOSUB  MP60            Hours*60
3956 1327A A70          A=A+B  W
3957 1327D 7A00         GOSUB  MP60            (hr*60 + min)*60
3958 13281 AF6          C=A    W
3959 13284 A7B          C=C+D  W               Hr*3600+min*60+sec
3960 13287 6E5A         GOTO   DECHEX          To HEX
3961                ****************************************************************
3962                ****************************************************************
3963                **
3964                ** Name:    MP60    -  Multiply A By 60
```

```
3965                **
3966                ** Category:   LOCAL
3967                **
3968                ** Purpose:
3969                **      Multiply A by 60.
3970                **
3971                ** Entry:
3972                **      A = BCD integer.
3973                **      DEC mode.
3974                **
3975                ** Exit:
3976                **      A = input * 60.
3977                **      DEC mode.
3978                **
3979                ** Calls:      None.
3980                **
3981                ** Uses.......
3982                **             A,C.
3983                **
3984                ** Stk lvls:   0
3985                **
3986                ** History:
3987                **
3988                **     Date     Programmer            Modification
3989                **   --------   ----------   ---------------------------------
3990                **   05/27/82   NM           Added documentation
3991                **
3992                ********************************************************************
3993                ********************************************************************
3994 1328B A74      MP60   A=A+A  W              2*a
3995 1328E AF6             C=A    W
3996 13291 A74             A=A+A  W              4*a
3997 13294 A7A             A=A+C  W              6*a in a
3998 13297 BF0             ASL    W              60*a
3999 1329A 01              RTN
4000                *
4001                ********************************************************************
4002                ********************************************************************
4003                **
4004                ** Name:    M306    -  Multiply Argument By 30.6
4005                **
4006                ** Category:   LOCAL
4007                **
4008                ** Purpose:
4009                **      Compute 30.6*B for time computations.
4010                **
4011                ** Entry:
4012                **      B = argument (BCD number).
4013                **      DEC mode.
4014                **
4015                ** Exit:
4016                **      C = argument * 30.6 (BCD number).
4017                **      DEC mode.
4018                **      P unaffected.
4019                **
```

```
4020                ** Calls:      None.
4021                **
4022                ** Uses.......
4023                **             B,C.
4024                **
4025                ** Stk lvls:   0
4026                **
4027                ** History:
4028                **
4029                **     Date     Programmer          Modification
4030                **   --------   ----------   ---------------------------------
4031                **   05/27/82   NM           Added documentation
4032                **
4033                ********************************************************************
4034                ********************************************************************
4035 1329C AF9   M306   C=B    W
4036 1329F C6          C=C+C  A
4037 132A1 C9          C=B+C  A              3*b
4038 132A3 D5          B=C    A
4039 132A5 C5          B=B+B  A              6*b
4040 132A7 F5          BSR    A              Int(.6*b)
4041 132A9 F2          CSL    A              30*b
4042 132AB C9          C=B+C  A              Int(30.6*b)
4043 132AD 01          RTN
4044                ********************************************************************
4045                ********************************************************************
4046                **
4047                ** Name:    SUM3    -  Compute Day# Of Start Of Year
4048                **
4049                ** Category:   LOCAL
4050                **
4051                ** Purpose:
4052                **      Compute day# of start of year for various date
4053                **      calculations.
4054                **
4055                ** Entry:
4056                **      A = Year (BCD number).
4057                **      DEC mode.
4058                **
4059                ** Exit:
4060                **      C = Result (day# since day 0).
4061                **      P=0.
4062                **      Carry clear.
4063                **      DEC mode.
4064                **
4065                ** Calls:      None.
4066                **
4067                ** Uses.......
4068                **             B,C,P.
4069                **
4070                ** Stk lvls:   0
4071                **
4072                ** Detail:
4073                **      A is untouched.
4074                **      Argument > 9999 is interpreted as -1.
```

```
4075                **      SUM3(-1)=-366 (10's complement).
4076                **
4077                ** Algorithm:
4078                **      SUM3(Y)=INT(Y*365.25)-INT(Y/100)+INT(Y/400)
4079                **
4080                ** History:
4081                **
4082                **     Date     Programmer            Modification
4083                **   --------   ----------   ---------------------------------
4084                **   05/27/82   NM           Added documentation
4085                **
4086                ********************************************************************
4087                ********************************************************************
4088 132AF AF2   SUM3   C=0    W
4089 132B2 23          P=     3
4090 132B4 A96         C=A    WP              Copy lower 4 digits of y
4091 132B7 20          P=     0
4092 132B9 972         ?A=C   W               Y<=9999
4093 132BC D0          GOYES  SUM310          Yes
4094 132BE 3366        LCHEX  0366
          30
4095 132C4 BFA         C=-C   W               Return -366
4096 132C7 03          RTNCC
4097 132C9 C2    SUM310 C=C+A  A
4098 132CB C2          C=C+A  A               3*A
4099 132CD AF5         B=C    W
4100 132D0 C5          B=B+B  A               6*A
4101 132D2 BF2         CSL    W               30*A
4102 132D5 A79         C=B+C  W               36*A
4103 132D8 BF2         CSL    W               360*A
4104 132DB A79         C=B+C  W               366*A
4105 132DE B72         C=C-A  W               365*A
4106 132E1 A75         B=B+B  W               12*A
4107 132E4 A75         B=B+B  W               24*A
4108 132E7 A78         B=B+A  W               25*A
4109 132EA BF5         BSR    W
4110 132ED F5          BSR    A               Int(A/4)
4111 132EF A79         C=B+C  W
4112 132F2 F5          BSR    A
4113 132F4 F5          BSR    A               Int(A/400)
4114 132F6 A79         C=B+C  W
4115 132F9 D8          B=A    A
4116 132FB F5          BSR    A
4117 132FD F5          BSR    A               Int(A/100)
4118 132FF B79         C=C-B  W               C=SUM3(A)
4119 13302 03          RTNCC
4120                ********************************************************************
4121                ********************************************************************
4122                **
4123                ** Name:(S) YMDDAY  -  Convert Year,month,day To Day#
4124                **
4125                ** Category:   TIME
4126                **
4127                ** Purpose:
4128                **      Convert date to absolute day#.
```

```
4129                ** 
4130                ** Entry:
4131                **      A = Year (BCD number).
4132                **      B = Month (BCD number).
4133                **      D = Day (BCD number).
4134                ** 
4135                ** Exit:
4136                **      A,B,C = Day# since day 0 (HEX).
4137                **      HEX mode.
4138                **      P=0.
4139                **      Carry clear.
4140                ** 
4141                ** Calls:      M306, SUM3, DECHEX (falls through)
4142                ** 
4143                ** Uses.......
4144                **              A,B,C,D,P
4145                ** 
4146                ** Stk lvls:   1
4147                ** 
4148                ** Detail:
4149                **      Day# is expressed relative to 1 January 0000.
4150                ** 
4151                ** Algorithm:
4152                ** 
4153                **  Define the following conditionally depending on
4154                **  the value of MONTH:
4155                ** 
4156                **          If MONTH < 3 then let M = MONTH + 13
4157                **                       and let Y = YEAR - 1.
4158                **          If MONTH >= 3 then let M = MONTH + 1
4159                **                       and let Y = YEAR.
4160                ** 
4161                **  Also define the following functions:
4162                ** 
4163                ** SUM3(Y) = int(Y * 365.25) - int(Y / 100) + int(Y / 400)
4164                **          = -366 if Y=-1
4165                ** M306(M) = int(M * 30.6001)
4166                ** 
4167                ** Mapping DATE to DAY NUMBER:
4168                ** 
4169                ** DAY#(MONTH,DAY,YEAR) = SUM3(Y) + M306(M) + DAY - 63
4170                ** 
4171                ** 
4172                ** History:
4173                ** 
4174                **     Date     Programmer          Modification
4175                **   --------   ----------   ---------------------------------
4176                **   05/27/82   NM           Added documentation
4177                ** 
4178                ********************************************************************
4179                ********************************************************************
4180 13304 05   =YMDDAY SETDEC
4181 13306 E5           B=B+1  A              Month+1
4182 13308 20           P=     0
4183 1330A D2           C=0    A
```

```
4184 1330C 304          LCHEX  4
4185 1330F 9E9          ?B>=C  B              Month+1 >= 4?
4186 13312 A0           GOYES  YMDD10         Yes, y=year, m=month+1
4187 13314 CC           A=A-1  A              No, y=year-1
4188 13316 3121         LCHEX  12
4189 1331A C1           B=B+C  A              M=month+13
4190 1331C 7C7F  YMDD10 GOSUB  M306           M306(M) to c
4191 13320 C3           D=D+C  A              Add to day
4192 13322 D2           C=0    A
4193 13324 3136         LCHEX  63
4194 13328 E3           D=D-C  A              Subtract 63
4195 1332A 718F         GOSUB  SUM3           SUM3(Y) to c
4196 1332E A7B          C=C+D  W              SUM3(Y)+M306(M)+day-63
4197 13331 64B9         GOTO   DECHEX
4198            ******************************************************************
4199            ******************************************************************
4200            **
4201            ** Name:(S) DAYYMD  -  Day# To Year, Month, Day
4202            **
4203            ** Category:   TIME
4204            **
4205            ** Purpose:
4206            **      Convert from absolute day# to date.
4207            **
4208            ** Entry:
4209            **      C = Day# since day 0 (HEX).
4210            **
4211            ** Exit:
4212            **      A = Year (BCD number).
4213            **      B = Month (BCD number).
4214            **      D = Day (BCD number).
4215            **
4216            ** Calls:      HEXDEC, ESTY0, SUM3, CHKY0, IDIV, M306, ASLW4
4217            **
4218            ** Uses.......
4219            **             A,B,C,D,P
4220            **
4221            ** Stk lvls:   1
4222            **
4223            ** Algorithm:
4224            **
4225            **  Define the following conditionally depending on
4226            **  the value of MONTH:
4227            **
4228            **         If MONTH < 3 then let M = MONTH + 13
4229            **                      and let Y = YEAR - 1.
4230            **         If MONTH >= 3 then let M = MONTH + 1
4231            **                      and let Y = YEAR.
4232            **
4233            **  Also define the following functions:
4234            **
4235            ** SUM3(Y) = int(Y * 365.25) - int(Y / 100) + int(Y / 400)
4236            **         = -366 if Y=-1
4237            ** M306(M) = int(M * 30.6001)
4238            **
```

```
4239                ** Mapping DAY NUMBER to DATE:
4240                **
4241                **   Calculate the value of Y0 as follows:
4242                **
4243                **     Y0 = int( [(DAY# + 63) - 121.5] / 365.2425)
4244                **   This is an approximation of the correct year.
4245                **
4246                ** # Now calculate M0 as follows:
4247                **     M0 = int( [(DAY# + 63) - SUM3(Y0)] / 30.6001)
4248                **   If this M0 is less than 4 then the year was one too
4249                **   high; therefore let Y0 = Y0 - 1 and recalculate M0
4250                **   using the new Y0 (ie Y0 := Y0 - 1 ; GO TO #).
4251                **
4252                **   Once a value for M0 greater than or equal to 4 is
4253                **   obtained, the values of MONTH, DAY, and YEAR are
4254                **   calculated as follows:
4255                **     DAY = [(DAY# + 63) - SUM3(Y0)] - M306(M0).
4256                **     If M0 >=14 then MONTH = M0 - 13 and YEAR = Y0 + 1.
4257                **     If M0 < 14 then MONTH = M0 - 1 and YEAR = Y0.
4258                **
4259                ** 360-day calendar is not done in this code.  Here is how to
4260                ** do it:
4261                **
4262                ** For 360 day calendar, the number of days between two
4263                ** dates is calculated as follows:
4264                **
4265                **    Let M1 = month of first date
4266                **    Let D1 = day of month of first date
4267                **    Let Y1 = year of first date
4268                **    Let M2 = month of second date
4269                **    Let D2 = day of month of second date
4270                **    Let Y2 = year of second date
4271                **
4272                **  Now make the following adjustments:
4273                **       If D1 >= 30 then
4274                **         begin
4275                **           D1 := 30;
4276                **           if D2 = 31 then D2 := 30
4277                **         end;
4278                **  Now compute:
4279                **    Delta-days = (Y2-Y1)*360  +  (M2-M1)*30  +  (D2-D1)
4280                **
4281                ** History:
4282                **
4283                **     Date     Programmer            Modification
4284                **    --------   ----------   ----------------------------------
4285                **    05/27/82   NM           Added documentation
4286                **
4287                ********************************************************************
4288                ********************************************************************
4289 13335 73B9 =DAYYMD GOSUB  HEXDEC          Convert day# to DEC
4290 13339 7E60         GOSUB  ESTY0          Estimate y0
4291 1333D 7E6F  DAYY10 GOSUB  SUM3           SUM3(Y0) to c
4292 13341 7A90         GOSUB  CHKY0          Check if y0 is correct
4293 13345 57F          GONC   DAYY10         Y0 was incremented.  try again.
```

```
4294 13348 3136          LCHEX  63            Prepare to compute m0
4295 1334C A79           C=B+C  W             Day#+63-SUM3(Y0)==q
4296 1334F AF7           D=C    W             Save q in d
4297 13352 AFE           ACEX   W             Position q for divide
4298 13355 2B            P=     11
4299 13357 BF3  DAYY30   DSL    W             Loop to sl q 5 times
4300 1335A 0C            P=P+1
4301 1335C 5AF           GONC   DAYY30
4302 1335F D7            D=C    A             Stash year
4303 13361 AF2           C=0    W
4304 13364 3510          LCHEX  306001        30.6001*10000
           0603
4305 1336C 8E00          GOSUBL =ASLW4        Q*10000
           00
4306 13372 7469          GOSUB  idiv          A/30.6001=m0
4307 13376 AF8           B=A    W
4308 13379 7F1F          GOSUB  M306
4309 1337D AF8           B=A    W             M0
4310 13380 AFA           A=C    W             M306(M0)
4311 13383 AF2           C=0    W
4312 13386 DB            C=D    A             Recover y0
4313 13388 AFE           ACEX   W             Y0 to A, M306(M0) to C
4314 1338B 2B            P=     11
4315 1338D BF7  DAYY40   DSR    W             Loop to recover q
4316 13390 0C            P=P+1
4317 13392 5AF           GONC   DAYY40
4318 13395 B73           D=D-C  W             Day
4319 13398 A6D           B=B-1  B             M0-1
4320 1339B 3121          LCHEX  12
4321 1339F 9ED           ?B<=C  B             M0<14?
4322 133A2 00            RTNYES               Yes, we are done
4323 133A4 E4            A=A+1  A             No, increment y0 for year
4324 133A6 B61           B=B-C  B             Month=m0-13
4325 133A9 01            RTN
4326            ****************************************************************
4327            ****************************************************************
4328            **
4329            ** Name:    ESTY0   -  Estimate Year From Day#
4330            **
4331            ** Category:   LOCAL
4332            **
4333            ** Purpose:
4334            **      Estimate year for use in day to date calculations.
4335            **
4336            ** Entry:
4337            **      C = Day# (DEC number since day 0).
4338            **      DEC mode.
4339            **
4340            ** Exit:
4341            **      D = C on input.
4342            **      A[A] = Estimate of Y0 (transformed year).
4343            **      DEC mode.
4344            **
4345            ** Calls:      IDIV
4346            **
```

```
4347                ** Uses.......
4348                **            A,B,C,D,P
4349                **
4350                ** Stk lvls:   0
4351                **
4352                ** NOTE:
4353                **      Value returned must be checked for validity in the
4354                **      calling routine by calling CHKYO.  This was done to
4355                **      save subroutine stack usage.
4356                **
4357                ** Algorithm:
4358                **      A=(day#+63-121.5).
4359                **      If A<0, then A=0; RTN.
4360                **      A=A/365.2425; RTN.
4361                **
4362                ** History:
4363                **
4364                **     Date     Programmer            Modification
4365                **   --------   ----------   ---------------------------------
4366                **   05/28/82   NM           Added documentation
4367                **
4368                ********************************************************************
4369                ********************************************************************
4370 133AB AF7    ESTYO  D=C    W              Save day#
4371 133AE AFA           A=C    W
4372 133B1 BF0           ASL    W              Day# * 10
4373 133B4 AF2           C=0    W
4374 133B7 20            P=     0
4375 133B9 3258          LCHEX  585            (121.5-63)*10
            5
4376 133BE B7A           A=A-C  W              (day#+63-121.5)*10
4377 133C1 570           GONC   ESTYOA         Go if no underflow
4378 133C4 AF0           A=0    W              Else use year=0
4379 133C7 01            RTN
4380 133C9 BF0    ESTYOA ASL    W
4381 133CC BF0           ASL    W
4382 133CF BF0           ASL    W              (day#+63-121.5)*10000
4383 133D2 3652          LCHEX  3652425        365.2425*10000
            4256
            3
4384 133DB 6EF8          GOTO   idiv           Compute estimate of year y0
4385                ********************************************************************
4386                ********************************************************************
4387                **
4388                ** Name:    CHKYO   -  Check Year Estimate For Validity
4389                **
4390                ** Category:   LOCAL
4391                **
4392                ** Purpose:
4393                **      Check year estimate for validity in day-date
4394                **      computations.
4395                **
4396                ** Entry:
4397                **      D = Day#  (BCD number).
4398                **      A[A] = Y0 (estimate made in ESTYO).
```

```
4399                ** C = SUM3(YO).
4400                ** DEC mode.
4401                **
4402                ** Exit:
4403                **      CARRY SET: Result was valid.
4404                **                 YO in A, Day# in D.
4405                **      CARRY CLEAR: Result was invalid.
4406                **                   YO-1 in A, Day# in D.
4407                **      P=0.
4408                **      B=SUM3(YO).
4409                **      C=00000000000000059.
4410                **
4411                ** Calls:      none.
4412                **
4413                ** Uses.......
4414                **             A,B,C,P.
4415                **
4416                ** Stk lvls:   0
4417                **
4418                ** NOTE:
4419                **      YO is considered in A[A], not A[W].  This doesn't
4420                **      matter except when YO=-1, in which case A returns the
4421                **      value 0000000000099999, not 9999999999999999.
4422                **
4423                ** Detail:
4424                **      Used by calling routine in conjunction with ESTYO and
4425                **      SUM3 to estimate year in day to date computations.
4426                **      Code was broken up this way to save subroutine levels.
4427                **
4428                ** Algorithm:
4429                **      If DAY#-SUM3(YO) <= 59, result is valid; RTNSC.
4430                **      Else A=A-1 W; RTNCC.
4431                **
4432                ** History:
4433                **
4434                **     Date     Programmer            Modification
4435                **   --------   ----------   --------------------------------
4436                **   05/28/82   NM           Added documentation
4437                **
4438                ********************************************************************
4439                ********************************************************************
4440 133DF BFA   CHKYO  C=-C   W
4441 133E2 A7B         C=C+D  W              Day#-SUM3(YO)
4442 133E5 AF5         B=C    W              Hold for compare
4443 133E8 AF2         C=0    W
4444 133EB 20          P=     0
4445 133ED 3195        LCHEX  59
4446 133F1 9F1         ?B>C   W              Day#-SUM3(YO)<=59?
4447 133F4 00          RTNYES                No.  we have valid yO.
4448 133F6 CC          A=A-1  A              Yes, decrement year
4449 133F8 03          RTNCC                 And try again
4450                ********************************************************************
4451                ********************************************************************
4452                **
4453                ** Name:    JD2DAY  -  Convert Julian Date To DAY#
```

```
4454                ;;
4455                ;; Category:   TIME
4456                ;;
4457                ;; Purpose:
4458                ;;      Convert Julian date to DAY#.  Used by SETDATE.
4459                ;;
4460                ;; Entry:
4461                ;;      A[W] = Year (BCD number).
4462                ;;      C = Day-of-year (BCD number).
4463                ;;
4464                ;; Exit:
4465                ;;      C = DAY# (HEX).
4466                ;;      HEX mode.
4467                ;;
4468                ;; Calls:      YMDDAY.
4469                ;;
4470                ;; Uses.......
4471                ;;              A,B,C,D,P
4472                ;;
4473                ;; Stk lvls:   1
4474                ;;
4475                ;; Algorithm:
4476                ;;      B=1; D=day-of-year; YMDDAY.
4477                ;;      (fool YMDDAY into Jan [day-of-year] YEAR.  For example,
4478                ;;       day-of-year 144 = Jan 144 YEAR.)
4479                ;;
4480                ;; History:
4481                ;;
4482                ;;     Date     Programmer             Modification
4483                ;;   --------   ----------   ---------------------------------
4484                ;;   05/28/82   NM           Added documentation
4485                ;;
4486                ;;*****************************************************************
4487                ;;*****************************************************************
4488 133FA AF7   JD2DAY D=C    W              Will use day# in january
4489 133FD AF1          B=0    W
4490 13400 B65          B=B+1  B              Indicate january
4491 13403 600F         GOTO   YMDDAY
4492                ;;*****************************************************************
4493                ;;*****************************************************************
4494                ;;
4495                ;; Name:(S) DAY2JD  -  Day# To Julian Date
4496                ;;
4497                ;; Category:   TIME
4498                ;;
4499                ;; Purpose:
4500                ;;      Convert day# (since 1 Jan 0000) to Julian date (year
4501                ;;      and day-in-year)
4502                ;;
4503                ;; Entry:
4504                ;;      C[W] = Day# (HEX days since day 0).
4505                ;;
4506                ;; Exit:
4507                ;;      A[W] = Year (BCD number).
4508                ;;      B,C = Day-of-year (BCD number).
```

```
4509                **      DEC mode.
4510                **
4511                ** Calls:      HEXDEC, ESTYO, SUM3, CHKYO.
4512                **
4513                ** Uses.......
4514                **             A,B,C,D,P.
4515                **
4516                ** Stk lvls:   1
4517                **
4518                ** Algorithm:
4519                **      Convert day# to DEC.
4520                **      Estimate YO.
4521                **   1: Compute SUM3(YO).
4522                **      CHKYO; if too high, decrement and goto 1.
4523                **      If SUM3(YO) <= 365 then goto 2.
4524                **      Day-in-year = SUM3(YO)-365.
4525                **      Year = YO+1.
4526                **      RTN.
4527                **   2: If year divisible by 100 then point at digit 2,
4528                **        else point at digit 0.
4529                **      If selected digit divisible by 4 then
4530                **          day-in-year=SUM3(YO)+1
4531                **        else
4532                **          day-in-year=SUM3(YO).
4533                **      Year = YO.
4534                **      RTN.
4535                **
4536                ** History:
4537                **
4538                **      Date      Programmer          Modification
4539                **    --------    ----------   ---------------------------------
4540                **    06/03/82    NM           Added documentation
4541                **
4542                ********************************************************************
4543                ********************************************************************
4544 13407 71E8 =DAY2JD GOSUB  HEXDEC          Convert day# to DEC
4545 1340B 7C9F        GOSUB  ESTYO           Estimate y0
4546 1340F 7C9E  DAY210 GOSUB  SUM3
4547 13413 78CF        GOSUB  CHKYO           YO correct?
4548 13417 57F         GONC   DAY210          No, try again
4549 1341A 3256        LCHEX  365
           3
4550 1341F 9BD         ?B<=C  X              1 jan or later?
4551 13422 B0          GOYES  DAY220          No
4552 13424 E1          B=B-C  A              Yes
4553 13426 AF9         C=B    W              This is day-in-year
4554 13429 E4          A=A+1  A              This is year
4555 1342B 01          RTN
4556 1342D 20    DAY220 P=      0             Mar 1 or later, check for leap
4557 1342F 96C         ?A#0   B              Year divisible by 100?
4558 13432 40          GOYES  DAY230          No, check for div 4
4559 13434 22          P=     2              Yes, check for div 400
4560 13436 303   DAY230 LCHEX  3
4561 13439 0E02        C=A&C  P
4562 1343D 90E         ?C#0   P              Is this leap year?
```

```
4563 13440 40          GOYES  DAY240        No
4564 13442 E5          B=B+1  A             Yes, add 1 to day-in-year
4565 13444 AF9  DAY240 C=B    W
4566 13447 01          RTN
4567 13449             END
```

```
 ACLC24  Abs   76665 #12B79 -  2219  2273  2570  2821
=ACTTMR  Abs   78249 #131A9 -  3619
=ADJA    Abs   75930 #1289A -  1096  2640  2829
 ADJAA1  Abs   77072 #12D10 -  2637  2565
 ADJAA2  Abs   77079 #12D17 -  2639  2569  2572
=ADJAAA  Abs   77064 #12D08 -  2635
=ADJN    Abs   75813 #12825 -   940  2693
=ADJNNN  Abs   77100 #12D2C -  2688
 ADSUB1  Abs   77176 #12D78 -  2761  2756
 ADSUB2  Abs   77194 #12D8A -  2767  2762
 ADSUB3  Abs   77197 #12D8D -  2768  2760  2764
 ADSUBT  Abs   77134 #12D4E -  2747  2638  2691
=AF      Abs   77292 #12DEC -  2917
 AF05    Abs   77374 #12E3E -  2941  2918
 AF07    Abs   77382 #12E46 -  2943  2940
 AF20    Abs   77408 #12E60 -  2951  2949
=ALMSRV  Abs   75133 #1257D -   191
 ALRM1   Abs  194329 #2F719 -    11  1371  3672
 ALRM6   Abs  194389 #2F755 -    11   747
 ARGERR  Ext                -  2699
 ARGSTA  Ext                -  2588
 ASLW4   Ext                -  4305
 ASRW3   Ext                -  3278
 BADAF   Abs   77277 #12DDD -  2876  2929
 CHKA20  Abs   75419 #1269B -   538   536
 CHKA25  Abs   75424 #126A0 -   540   472
 CHKA30  Abs   75427 #126A3 -   541   539
 CHKA40  Abs   75435 #126AB -   544   484
 CHKAF   Abs   75403 #1268B -   532  2928
 CHKY0   Abs   78815 #133DF -  4440  4292  4547
 CLKPRS  Ext                -  2562  2634  2687  2817
 CLKU10  Abs   75508 #126F4 -   740   733
 CLKU15  Abs   75524 #12704 -   745   790
 CLKU20  Abs   75538 #12712 -   748   765
 CLKU30  Abs   75570 #12732 -   759   753
 CLKU40  Abs   75578 #1273A -   762   751   755   758   760
 CLKU54  Abs   75671 #12797 -   791   777
 CLKU55  Abs   75674 #1279A -   792   775
 CLKU60  Abs   75718 #127C6 -   811   799   801
 CLKU70  Abs   75750 #127E6 -   820   822
 CLKUPD  Abs   75456 #126C0 -   725   330   962  1330
 CLRFRC  Ext                -  2282
=CMPT    Abs   75186 #125B2 -   302   198   373  1161  1269  1315  2272  3509
                              3562
 CMPT20  Abs   75235 #125E3 -   318   316
 CMPT30  Abs   75250 #125F2 -   323   321
 CMPTE   Abs   75892 #12874 -   998   947   950
=CMPTIM  Abs   75277 #1260D -   373  2333  2389  2447
 COLN10  Abs   76925 #12C7D -  2511  2517
 COLN12  Abs   76897 #12C61 -  2501  2392  2451
 COMP10  Abs   75327 #1263F -   460   457
 COMP20  Abs   75364 #12664 -   471   459
 COMP40  Abs   75383 #12677 -   478   475
 COMP50  Abs   75394 #12682 -   482   479
 COMPAF  Abs   75292 #1261C -   449  1171
```

```
=DATE    Abs   76756 #12BD4 -  2332
=DATE$   Abs   76858 #12C3A -  2446
 DAY210  Abs   78863 #1340F -  4546  4548
 DAY220  Abs   78893 #1342D -  4556  4551
 DAY230  Abs   78902 #13436 -  4560  4558
 DAY240  Abs   78916 #13444 -  4565  4563
=DAY2JD  Abs   78855 #13407 -  4544  2336
 DAYY10  Abs   78653 #1333D -  4291  4293
 DAYY30  Abs   78679 #13357 -  4299  4301
 DAYY40  Abs   78733 #1338D -  4315  4317
=DAYYMD  Abs   78645 #13335 -  4289  2450  3329  3442
 DCHXW   Ext                -  2586
 DECD10  Abs   77834 #1300A -  3283  3281
 DECD20  Abs   77858 #13022 -  3291  3289
 DECDAT  Abs   77766 #12FC6 -  3263  2819
 DECHEX  Abs   77030 #12CE6 -  2586  2927  3035  3746  3960  4197
 DECT10  Abs   77470 #12E9E -  3031  3029
 DECTIM  Abs   77412 #12E64 -  3015  2564  2636  2689
 DRANGE  Ext                -  3147
 DV2-12  Ext                -  2281
 DV60    Abs   78440 #13268 -  3909  3904  3906
 ESTY0   Abs   78763 #133AB -  4370  4290  4545
 ESTY0A  Abs   78793 #133C9 -  4380  4377
 EXAC10  Abs   75989 #128D5 -  1171  1167
 EXAC20  Abs   76002 #128E2 -  1175  1172
 EXAC30  Abs   76009 #128E9 -  1177  1170  1173
=EXACT   Abs   75952 #128B0 -  1161  2874
=EXACTT  Abs   77270 #12DD6 -  2874
 EXPEXC  Ext                -  2589
 EXPR    Ext                -  2402
 FIXDC   Ext                -  2561  2633  2686  2816
 FLOAT   Ext                -  1859
=FLOATA  Abs   76373 #12A55 -  1859  2277  2341  2946
 FNRTN1  Ext                -  2287
=FROMDT  Abs   78395 #1323B -  3852
 GETA10  Abs   76258 #129E2 -  1585  1431
 GETA20  Abs   76264 #129E8 -  1587  1675
 GETAF   Abs   76234 #129CA -  1579   313   455   463   798  2936  2942
 GETD10  Abs   77731 #12FA3 -  3204  3208
 GETD20  Abs   77744 #12FB0 -  3209  3205
 GETD30  Abs   77754 #12FBA -  3212  3214
 GETDFL  Abs   77764 #12FC4 -  3216  3199
 GETDIG  Abs   77707 #12F8B -  3196  3040  3044  3049  3304  3308  3316  3322
 GETIRQ  Abs   76212 #129B4 -  1504   304   743
 GETL12  Abs   76295 #12A07 -  1670  1505  1745
 GETLAF  Abs   76288 #12A00 -  1659   309   323   783   958
 GETLST  Abs   76327 #12A27 -  1744   450   467   786   954
 GETOFS  Abs   76170 #1298A -  1425   452   951
 GETPND  Abs   76349 #12A3D -  1820   192   767  1382
 GETTIM  Abs   75082 #1254A -   125   194   302
 GETTM2  Abs   75116 #1256C -   135   133
 GTFLAG  Ext                -   728  1163
 HEXDEC  Abs   77036 #12CEC -  2587  2275  2944  3902  4289  4544
=HMSSEC  Abs   78452 #13274 -  3954  3052
 HXDCW   Ext                -  2587
```

```
 IDIV    Ext                  -  2584
=INITCL  Abs   78113 #13121 -  3497
=Invarg  Abs   77128 #12D48 -  2699
 JD2DAY  Abs   78842 #133FA -  4488  3296
 LC708   Abs   75918 #1288E -  1039   730   999  1003
 LC9999  Abs   75782 #12806 -   863   773  1321  2761
 M306    Abs   78492 #1329C -  4035  4190  4308
 MP2-12  Ext                  -  3025  3741
 MP60    Abs   78475 #1328B -  3994  3955  3957
 MPY     Ext                  -  2585
 MfErr   Ext                  -  2878
 NEXTst  Ext                  -  2830
 NXTIRQ  Abs  194317 #2F70D -    11  1504  1538  3497  3500
 PNDALM  Abs  194401 #2F761 -    11  1820  1855
 POP1S   Ext                  -  3106
 PRSD10  Abs   77947 #1307B -  3321  3313
 PRSD20  Abs   77960 #13088 -  3325  3320
 PRSDAT  Abs   77877 #13035 -  3298  3268
 PRSTIM  Abs   77488 #12EB0 -  3036  3020
 PUTA10  Abs   76127 #1295F -  1372  1376
 PUTA20  Abs   76143 #1296F -  1378  1373
 PUTAF   Abs   76275 #129F3 -  1622  1176  1222  2938
=PUTALM  Abs   76114 #12952 -  1368  1268  1329
 PUTIRQ  Abs   76223 #129BF -  1538   813
 PUTL12  Abs   76321 #12A21 -  1709  1469  1539  1778
 PUTLAF  Abs   76314 #12A1A -  1708   326   785   960  1180  2933
 PUTLST  Abs   76338 #12A32 -  1777   788   957  1179  2934
 PUTOFO  Abs   76198 #129A6 -  1467  1177  2935
 PUTOFS  Abs   76201 #129A9 -  1468   953
=PUTPND  Abs   76361 #12A49 -  1855   771  1385
 RANG10  Abs   78016 #130C0 -  3379  3376
 RANG20  Abs   78035 #130D3 -  3384  3382
 RANGYR  Abs   78000 #130B0 -  3373  3279  3319
=RC01    Abs   76486 #12AC6 -  2032   199  3578
 RCDOD1  Abs   76419 #12A83 -  1944  2286  2393  3017  3265
 RCTT10  Abs   78284 #131CC -  3674  3676
=RCTTM1  Abs   78258 #131B2 -  3667
=RESETC  Abs   76024 #128F8 -  1219
 RJUS05  Abs   76537 #12AF9 -  2094  2092
 RJUS07  Abs   76545 #12B01 -  2097  2090
 RJUS10  Abs   76576 #12B20 -  2107  2111
 RJUS20  Abs   76591 #12B2F -  2112  2109
 RJUS30  Abs   76603 #12B3B -  2116  2106
 RJUS99  Abs   76610 #12B42 -  2121  2102
=RJUST   Abs   76514 #12AE2 -  2086  2925  3033  3275  3744
 RJUSZR  Abs   76605 #12B3D -  2118  2099
 RTNCC   Ext                  -  2873
 SOCNEG  Abs   75799 #12817 -   869   314  2943
 SCREXO  Abs  194881 #2F941 -    11  3559  3575
 SCRTCH  Abs  194817 #2F901 -    11  1991  2032
=SEC2TK  Abs   78339 #13203 -  3737
=SECHMS  Abs   78418 #13252 -  3902  2391  3448
 SET10   Abs   75845 #12845 -   950   945
 SET20   Abs   75849 #12849 -   951   949
 SET30   Abs   75860 #12854 -   954  1102
```

```
 SETA10  Abs   76079 #1292F -  1320  3690
 SETA20  Abs   76100 #12944 -  1327  1325
=SETALM  Abs   76045 #1290D -  1267
=SETALR  Abs   76055 #12917 -  1312  3620
=SETDAT  Abs   77212 #12D9C -  2818
 SETF10  Abs   77315 #12E03 -  2925  2923
 SETI10  Abs   76975 #12CAF -  2570  2567
=SETIME  Abs   75813 #12825 -   939  2581
=SETTIM  Abs   76954 #12C9A -  2563
=SETTMO  Abs   78168 #13158 -  3557
 SF12?   Abs   75164 #1259C -   233   193   772
 SF12?1  Abs   75181 #125AD -   240   237
 SFLAG?  Ext                -  3565
 SFLAGC  Ext                -  1225
=ST01    Abs   76460 #12AAC -  1991   197  3558
 STDOD1  Abs   76380 #12A5C -  1894  2271  2332  2388  2446  2931  2941  3015
                            3263
 STM100  Abs   78224 #13190 -  3572  3567
=STMEND  Abs   77254 #12DC6 -  2830  2582  2641  2694  2875
 STOPDC  Ext                -  2872
 STRHDR  Ext                -  2398
 SUM3    Abs   78511 #132AF -  4088  4195  4291  4546
 SUM310  Abs   78537 #132C9 -  4097  4093
 TDPA10  Abs   77647 #12F4F -  3136  3131
 TDPA20  Abs   77650 #12F52 -  3137  3128  3151
 TDPA30  Abs   77661 #12F5D -  3141  3134
 TDPA40  Abs   77679 #12F6F -  3147  3143
 TDPA50  Abs   77692 #12F7C -  3150  3146
 TDPA60  Abs   77699 #12F83 -  3152  3139
 TDPARS  Abs   77553 #12EF1 -  3105  3037  3299
 TIM$20  Abs   76819 #12C13 -  2393  2456
 TIMAF   Abs  194439 #2F787 -    11  1579  1622  3500
=TIME    Abs   76685 #12B8D -  2271
=TIME$   Abs   76799 #12BFF -  2388
 TIME10  Abs   76744 #12BC8 -  2286  2343  2951
 TIMER2  Abs  189176 #2E2F8 -    11   126   817  3502
 TIMERR  Abs   77126 #12D46 -  2698  2093  2637  2639  2690  2692  2820
 TIMLAF  Abs  194427 #2F77B -    11  1659  1708
 TIMLST  Abs  194415 #2F76F -    11  1744  1777
 TIMOFS  Abs  194403 #2F763 -    11  1425  1468
=TIMRND  Abs   75281 #12611 -   374  3438
=TODT    Abs   78377 #13229 -  3809  2334  2390  2448  3439
 TRUN10  Abs   76655 #12B6F -  2180  2177
 TRUN20  Abs   76657 #12B71 -  2181  2171  2174
=TRUNCC  Abs   76618 #12B4A -  2168  2283  3031  3742
 WIPOUT  Ext                -  3501
 WRTTMR  Ext                -  3508
 YMDD10  Abs   78620 #1331C -  4190  4186
=YMDDAY  Abs   78596 #13304 -  4180  3327  4491
=YMDH01  Abs   78053 #130E5 -  3439
 YMDH10  Abs   78090 #1310A -  3450  2501  3444
 YMDH20  Abs   78093 #1310D -  3451  3325  3332
=YMDHMS  Abs   78043 #130DB -  3437
 argsta  Abs   77042 #12CF2 -  2588  2919  3021  3269
 cmpt    Abs   78162 #13152 -  3509  2932  3016  3264  3437  3670
```

```
 eAF      Ext                    -  2877
=expexc   Abs    77048 #12CF8    -  2589  2563  2635  2688  2818
 flCLOC   Ext                    -   727
 flCTON   Ext                    -  3564
 flEXAC   Ext                    -  1162  1224
 idiv     Abs    77018 #12CDA    -  2584   318   482   807  1002  1322  2274  2575
                                    2825  3684  3813  3912  4306  4384
 mpy      Abs    77024 #12CE0    -  2585   460   464   805  2577  2822  3856
 seta10   Abs    78333 #131FD    -  3690  3574
 setalr   Abs    78252 #131AC    -  3620
```

Input Parameters

  Source file name is MN&TM::MS

  Listing file name is MN/TM:TI:ML::-1

  Object file name is MN%TM:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                      TITLE  Flags, Traps, and Modes<831216.1456>
 2 13449                ABS    #13449
 3               *
 4               *  PPPP   M   M   &     FFFFF  L       GGG
 5               *  P   P  MM MM  & &    F      L      G   G
 6               *  P   P  M M M  & &    F      L      G
 7               *  PPPP   M M M   &     FFFF   L      G GGG
 8               *  P      M   M  & & &  F      L      G   G
 9               *  P      M   M  &  &   F      L      G   G
10               *  P      M   M   && &  F      LLLLL   GGG
11               *
12               ********************************************************************
13               ********************************************************************
14               **
15               ** Name:    SFLAG   -  User Set Flags
16               **
17               ** Category:   STEXEC
18               **
19               ** Purpose:
20               **      ALL:  sets all user flags (0 to 63)
21               **      MATH: sets all IEEE exception flags (-4 to -8)
22               **      otherwise: sets the flag(s) specified by the rounded
23               **                 value(s) of a flag number list.
24               **      in all cases, the annunciators are updated.
25               **
26               ** Entry:
27               **      D0 -- program counter, points to keyword or beginning
28               **            of flag number list
29               **      HEXMODE
30               **
31               ** Exit:
32               **      appropriate flags set
33               **      all annunciators updated
34               **
35               ** Calls:      expex-,FLALL,FLMATH,FLGNB?,FLGS/C,RNDAHX,UPDANN,
36               **             exits through NXTSTM
37               **
38               ** Uses.......
39               **  Exclusive of expex-:
40               **             CPU: A,B(S,A),C(5-0),D(A),P,SB,XM,D0,D1,sSTAT
41               **             RAM: ANNAD1-4,FLGREG,SYSFLG
42               **
43               ** Stk lvls:   ALL/MATH: 2
44               **             flag # list: max(4,1+EXPEX-)
45               **
46               ** NOTE:
47               **      Flag numbers must be in the range [-32,63].
48               **
49               ** History:
50               **
51               **    Date     Programmer            Modification
52               **  --------   ----------  ---------------------------------
53               **  06/11/82      PM        Documented routine
54               **  08/06/82      PM        Updated annunc. after each flag
55               **  01/05/83      PM        Revised documentation
```

```
                **
                ********************************************************************
                ********************************************************************
 13449 0000     0       REL(5) =SFLGDC          (same as FIXDC)
 1344E 0000     0       REL(5) =SFLGP
 13453 7380 =SFLAG  GOSUB  FLALL           test for ALL token
 13457 431          GOC    SFLAG0
 1345A A7C          A=A-1  W               ALL token: set all user flags
 1345D 1507  anxtsW DATO=A W
 13461 148   anxtsB DATO=A B
 13464 7901         GOSUB  UPDANN          update annunciators
 13468 5D6          GONC   nxtstm          B.E.T.
 1346B 7380  SFLAG0 GOSUB  FLMATH          test for MATH token
 1346F 4D0          GOC    SFLAG2
 13472 318F         LCHEX  F8              MATH token: set flags -4 to -8
 13476 0E6E         A=A!C  B
 1347A 56E          GONC   anxtsB          B.E.T.
 1347D 7962  SFLAG2 GOSUB  expex-          flag number list
 13481 850          ST=1   =sSTAT
 13484 6D20         GOTO   XFLAG3
                ********************************************************************
                ********************************************************************
                **
                ** Name:    CFLAG   -  User clear flags
                **
                ** Category:   STEXEC
                **
                ** Purpose:
                **      Clears the flag(s) specified by a keyword or a flag
                **      number list.  See SFLAG documentation for more
                **      information.
                **
                ********************************************************************
                ********************************************************************
 13488 0000     0       REL(5) =SFLGDC
 1348D 0000     0       REL(5) =SFLGP
 13492 7440 =CFLAG  GOSUB  FLALL           test for ALL token
 13496 56C          GONC   anxtsW          ALL token: clear all user flags
 13499 7550         GOSUB  FLMATH          test for MATH token
 1349D 4D0          GOC    CFLAG2
 134A0 3170         LCHEX  07              MATH token: clear flags -4 to -8
 134A4 0E66         A=A&C  B
 134A8 58B          GONC   anxtsB          B.E.T.
 134AB 7B32  CFLAG2 GOSUB  expex-          flag nbr list
 134AF 840          ST=0   =sSTAT
 134B2 7512  XFLAG3 GOSUB  RNDAHX          pop, test, round and convert to hex
 134B6 7202         GOSUB  FLGNB?          get flag if nbr in range, else err
 134BA 7061         GOSUB  FLGS/C          set/clear flag for sSTAT=1,0
 134BE 17F          D1=D1+ 16                set/clearing each flag
 134C1 133          AD1EX
 134C4 131          D1=A
```

```
107 134C7 1B00          DO=(5) =FORSTK
          000
108 134CE 146           C=DATO A
109 134D1 8B2           ?A<C   A                more arguments?
110 134D4 ED            GOYES  XFLAG3
111 134D6 6000  nxtstm  GOTO   =STMEND          To NXTSTM
112               ****************************************************************
113               ****************************************************************
114               **
115               ** Name:    FLALL   -  Tests for ALL token
116               **
117               ** Category:   LOCAL
118               **
119               ** Purpose:
120               **      Tests for ALL token.
121               **
122               ** Entry:
123               **      P      =  0
124               **      DO --- Program counter
125               **
126               ** Exit:
127               **      P      =  0
128               **      Carry=set:   no match
129               **                   DO unchanged
130               **      Carry=clear: match
131               **                   DO points to FLGREG
132               **                   A=0
133               **
134               ** Calls:      nothing
135               **
136               ** Uses.......
137               **  Inclusive: A,C(B),DO
138               **
139               ** Stk lvls:   0
140               **
141               ** History:
142               **
143               **    Date     Programmer              Modification
144               **  --------   ----------   ----------------------------------
145               **  08/23/82      PM         Documented routine
146               **  01/05/83      PM         Reviewed documentation
147               **
148               ****************************************************************
149               ****************************************************************
150 134DA 14A   FLALL  A=DATO B                get token
151 134DD 3100         LC(2)  =tALL            load ALL token
152 134E1 966          ?A#C   B
153 134E4 00           RTNYES                  carry set: no match
154 134E6 1B00         DO=(5) =FLGREG
          000
155 134ED AF0          A=0    W
156 134F0 01           RTN                     carry clear: match
157               ****************************************************************
158               ****************************************************************
159               **
```

```
160                ** Name:    FLMATH  -  Tests for MATH token
161                **
162                ** Category:   LOCAL
163                **
164                ** Purpose:
165                **      Tests for MATH token.
166                **
167                ** Entry:
168                **      P      =  0
169                **      D0 --- Program counter
170                **
171                ** Exit:
172                **      Carry=set:   no match
173                **                   D0 unchanged
174                **                   P=5
175                **      Carry=clear: match
176                **                   D0 points to SYSFLG
177                **                   P=0
178                **                   A(B) contains flags -1 thru -8
179                **
180                ** Calls:      nothing
181                **
182                ** Uses.......
183                **  Inclusive: A(5-0),C(5-0),P,D0
184                **
185                ** Stk lvls:   0
186                **
187                ** History:
188                **
189                **    Date     Programmer              Modification
190                **  --------   ----------   ---------------------------------
191                **  08/23/82      PM         Documented routine
192                **  01/05/83      PM         Reviewed documentation
193                **
194                ********************************************************************
195                ********************************************************************
196 134F2 15A5  FLMATH A=DAT0 6               get token
197 134F6 3500        LC(6)  =tMATH          load MATH token
          0000
198 134FE 25          P=     5
199 13500 916         ?A#C   WP
200 13503 00          RTNYES                 carry set: no match
201 13505 1B00        D0=(5) =SYSFLG
          000
202 1350C 14A         A=DAT0 B
203 1350F 20          P=     0
204 13511 01          RTN                    carry clear: match
205                ********************************************************************
206                ********************************************************************
207                **
208                ** Name:    FLAG    -  User's flag function
209                **
210                ** Category:   FNEXEC
211                **
212                ** Purpose:
```

```
213                ** Returns flag value of specified flag.  If an
214                ** optional new flag value is included, sets the
215                ** flag to that new value and returns the previous
216                ** flag value.
217                **
218                ** Entry:
219                **      flag # on top of math stack
220                **      HEXMODE
221                **
222                ** Exit:
223                **      value on top of math stack
224                **      optionally, flag set to new value
225                **
226                ** Calls:       FLGRG?,FLGUP?,PARMCT,POP1N+,RNDAHX
227                **              exits through BABBAJ
228                **
229                ** Uses.......
230                **  Exclusive of uRES12: CPU: A,B,C,D(S,A),P,D0,D1,SB,XM,sSTAT
231                **                       RAM: ANNAD1-4,FLGREG,SYSFLG
232                **                       unless fatal error
233                **
234                ** Stk lvls:   max(4,2+uRES12)
235                **
236                ** History:
237                **
238                **    Date      Programmer              Modification
239                **  --------    ----------   ----------------------------------
240                **  06/11/82       PM         Documented routine
241                **  01/05/83       PM         Revised documentation
242                **
243                ********************************************************************
244                ********************************************************************
245 13513 8812              NIBHEX 8812
246 13517 7140 =FLAG       GOSUB  PARMCT          set sSTAT according to # parms
247 1351B 4C1              GOC    NOFVAL
248 1351E 8E00             GOSUBL =POP1N+         pop new value, test for data
           00
249 13524 413              GOC    cnflct            type, signaling NaN
250 13527 17F              D1=D1+ 16
251 1352A AC3              D=0    S               save new value in D(S)
252 1352D AC0              A=0    S
253 13530 978              ?A=0   W
254 13533 50               GOYES  NOFVAL
255 13535 B47              D=D+1  S               nonzero: use 1
256 13538 7F81  NOFVAL     GOSUB  RNDAHX          pop, test, round, convert flag #
257 1353C 7471             GOSUB  FLGRG?          get flag if nbr in range, else err
258 13540 0E26             A=A&C  XS              perform logical "AND" operation
259 13544 AD0              A=0    M               clear mantissa
260 13547 F0               ASL    A               shift into A(M). A(M)=0 iff flg cl
261 13549 79D0             GOSUB  FLGUP?          update flag, annunciators?
262 1354D BD8              A=-A   M               if flg was clr, cy=clr, else =set
263 13550 8C00             GOLONG =BABBAJ         determine result, push on stack
           00
264 13556 8C00  cnflct     GOLONG =CNFLCT
           00
```

```
                ****************************************************************
                ****************************************************************
                **
                ** Name:    PARMCT  -  Saves parameter count in sSTAT
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Status bit sSTAT set according to C(S)
                **
                ** Entry:
                **      C(S)
                **      HEXMODE
                **
                ** Exit:
                **      sSTAT = 0 if C(S)=1
                **              1 otherwise
                **
                ** Calls:      nothing
                **
                ** Uses.......
                **  Inclusive: C(S),sSTAT
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ---------------------------------
                **  01/05/83      PM         Documented routine
                **
                ****************************************************************
                ****************************************************************
 1355C 840    PARMCT ST=0   =sSTAT
 1355F A4E             C=C-1  S
 13562 A4E             C=C-1  S
 13565 400             RTNC
 13568 850             ST=1   =sSTAT
 1356B 01              RTN
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) UPDANN  -  Update Annunciator
                ** Name:    UPDANX  -  Update Annunciator
                **
                ** Category:   DSPUTL
                **
                ** Purpose:
                **      Updates annunciators corresponding to user and system
                **      flags.
                **
                ** Entry:
                **      user and system flags
                **      HEXMODE
                **
```

```
320             ** Exit:
321             **      appropriate annunciator(s) turned on/off
322             **      Carry=Clear
323             **      P=0
324             **      HEXMODE
325             **
326             ** Calls:      DBLUP,SNGLUP,UPDAN1
327             **
328             ** Uses.......
329             **  Inclusive: CPU: A(B),B(A),C(A),D0,P
330             **             RAM: ANNAD1-4
331             **
332             ** Stk lvls:   1
333             **
334             ** History:
335             **
336             **    Date      Programmer              Modification
337             **  --------    ----------   ------------------------------------
338             **  06/11/82      PM         Documented routine
339             **  10/04/82      PM         Changed for annunciator revision
340             **  01/05/83      PM         Revised documentation
341             **
342             ********************************************************************
343             ********************************************************************
344 1356D 1502 =UPDANX DAT0=A XS
345 13571 20   =UPDANN P=     0
346 13573 1B00          D0=(5) (=FLGREG)-2    read some flags
            000
347 1357A 146           C=DAT0 A
348 1357D D5            B=C    A              save flags in B(A)
349 1357F 34E7          LCHEX  7E17E          masks for following calls
            1E7
350 13586 7E30          GOSUB  UPDAN1         AC annunciator
351 1358A 1A00          D0=(4) =ANNAD4
            00
352 13590 7E30          GOSUB  DBLUP          ALARM,PRGM,SUSP,CALC annunciators
353 13594 F6            CSR    A
354 13596 F5            BSR    A
355 13598 1A00          D0=(4) (=ANNAD2)+1
            00
356 1359E 7230          GOSUB  SNGLUP         BAT annunciator
357 135A2 F5            BSR    A
358 135A4 C5            B=B+B  A              shift user flags left 3 bits
359 135A6 C5            B=B+B  A
360 135A8 C5            B=B+B  A
361 135AA 1A00          D0=(4) =ANNAD3
            00
362 135B0 7E10          GOSUB  DBLUP          user flags 0-4
363 135B4 F6            CSR    A
364 135B6 1A00          D0=(4) (=FLGREG)-14   read system set/test/clear flags
            00
365 135BC 14A           A=DAT0 B
366 135BF A88           B=A    P
367 135C2 A05           B=B+B  P              shift flags left one bit
368 135C5 309           LCHEX  9              USER,RAD annunciators
```

```
369 135C8 1A00  UPDAN1 DO=(4) (=ANNAD1)+1
          00
370 135CE 6700         GOTO   SNGLP+
371               ****************************************************************
372               ****************************************************************
373               **
374               ** Name:     DBLUP   -  Turn annunc. on/off
375               ** Name:     SNGLUP  -  Turn annunc. on/off
376               ** Name:     SNGLP+  -  Turn annunc. on/off
377               **
378               ** Category:   LOCAL
379               **
380               ** Purpose:
381               **      Turns annunciator(s) off/on according to whether
382               **      corresponding flag(s) is cleared/set.
383               **
384               ** Entry(SNGLP+):
385               **      B(WP) - contains flags of interest
386               **      C(WP) - 1's complement of flag mask
387               **      P ---- points to position used by A,B,C
388               **      DO --- points to nibble containing annunciator
389               **      HEXMODE
390               **
391               ** Exit:
392               **      annunciator turned off/on
393               **      Carry=Clear
394               **      P=0
395               **      HEXMODE
396               **
397               ** Calls:      nothing
398               **
399               ** Uses.......
400               **  Inclusive: A(WP),C(A),DATO,P,SB
401               **
402               ** Stk lvls:   0
403               **
404               ** History:
405               **
406               **    Date      Programmer              Modification
407               **  --------    ----------   ----------------------------------
408               **  06/14/82       PM         Documented routine
409               **  11/12/82       PM         Packed as per M.B.'s suggestions
410               **  01/05/83       PM         Revised documentation
411               **
412               ****************************************************************
413               ****************************************************************
414 135D2 21    DBLUP  P=     1
415 135D4 F6    SNGLUP CSR    A
416 135D6 1521  SNGLP+ A=DATO WP
417 135DA 0E16         A=A&C  WP              turn annunc. off
418 135DE B9E          C=-C-1 WP
419 135E1 0E15         C=C&B  WP
420 135E5 0E1E         A=A!C  WP              turn annun. on if flag set
421 135E9 1501         DATO=A WP
422 135ED 20           P=     0
```

```
 423 135EF 03          RTNCC                   return with carry clear
 424            ****************************************************************
 425            ****************************************************************
 426            **
 427            ** Name:(S) SFLAGS  -  Sets system flag
 428            **
 429            ** Category:   GENUTL
 430            **
 431            ** Purpose:
 432            **      Sets a system flag and updates annunciators
 433            **
 434            ** Entry:
 435            **      C(B) -- hex flag number (e.g. load FF for -1)
 436            **      HEXMODE
 437            **      P=0
 438            **
 439            ** Exit:
 440            **      specified flag set
 441            **      any corresponding annunciator turned on
 442            **      Carry=Clear
 443            **      D(A) - Set to DO
 444            **      HEXMODE
 445            **      P=0
 446            **
 447            ** Calls:      GTFLAG,UPDANX
 448            **
 449            ** Uses.......
 450            **  Inclusive: CPU: A(A),B(A),C(15,5-0),D(A),P
 451            **             RAM: ANNAD1-4,SYSFLG
 452            **
 453            ** Stk lvls:   2
 454            **
 455            ** History:
 456            **
 457            **    Date     Programmer              Modification
 458            **  --------   ----------   ---------------------------------
 459            **  06/11/82      PM         Documented routine
 460            **  04/11/83      PM         Revised documentation
 461            **
 462            ****************************************************************
 463            ****************************************************************
 464 135F1 D9    SFLG*B C=B    A
 465 135F3 890  =SFLAG* ?P=    0
 466 135F6 B0           GOYES  SFLAGC
 467 135F8 20           P=     0
 468 135FA 7060 =SFLAGS GOSUB  GTFLAG          Set flag
 469 135FE 5E3          GONC   SYSFLS          B.E.T.
 470            ****************************************************************
 471            ****************************************************************
 472            **
 473            ** Name:(S) SFLAGC  -  Clears system flag
 474            **
 475            ** Category:   GENUTL
 476            **
 477            ** Purpose:
```

```
478             **      Cleares a system flag and updates annunciators
479             **
480             ** Entry:
481             **      C(B) -- hex flag number (e.g. load FF for -1)
482             **      HEXMODE
483             **      P=0
484             **
485             ** Exit:
486             **      specified flag cleared
487             **      any corresponding annunciator turned on
488             **      Carry=Clear
489             **      D(A) - Set to D0
490             **      HEXMODE
491             **      P=0
492             **
493             ** Calls:       GTFLAG,UPDANX
494             **
495             ** Uses.......
496             **  Inclusive: CPU: A(A),B(A),C(15,5-0),D(A),P
497             **             RAM: ANNAD1-4,SYSFLG
498             **
499             ** Stk lvls:   2
500             **
501             ** History:
502             **
503             **    Date      Programmer              Modification
504             **  --------    ----------   -----------------------------------
505             **  06/11/82      PM         Documented routine
506             **  04/11/83      PM         Revised documentation
507             **
508             ********************************************************************
509             ********************************************************************
510 13601 7950 =SFLAGC GOSUB  GTFLAG          Clear flag
511 13605 5E2         GONC   SYSFLC          B.E.T.
512             ********************************************************************
513             ********************************************************************
514             **
515             ** Name:(S) SFLAGT  -  Toggles system flag
516             **
517             ** Category:   GENUTL
518             **
519             ** Purpose:
520             **      Toggles a system flag and updates annunciators
521             **
522             ** Entry:
523             **      C(B) -- hex flag number (e.g. load FF for -1)
524             **      HEXMODE
525             **      P=0
526             **
527             ** Exit:
528             **      specified flag toggled
529             **      any corresponding annunciator turned on
530             **      Carry=Set if flag previously set
531             **      Carry=Clear if flag previously cleared
532             **      D(A) - Set to D0
```

```
533            **      HEXMODE
534            **      P=0
535            **
536            ** Calls:      GTFLAG,SYSFLC,UPDANX
537            **
538            ** Uses.......
539            **  Inclusive: CPU: A(A),B(A),C(15,5-0),D(A),P
540            **             RAM: ANNAD1-4
541            **
542            ** Stk lvls:   3
543            **
544            ** History:
545            **
546            **    Date     Programmer            Modification
547            **  --------   ----------   ---------------------------------
548            **  06/11/82      PM         Documented routine
549            **  04/11/83      PM         Revised documentation
550            **
551            ********************************************************************
552            ********************************************************************
553 13608 7250 =SFLAGT GOSUB  GTFLAG         Toggle flag
554 1360C AA8          B=A    XS
555 1360F 0E21         B=B&C  XS
556 13613 929          ?B=0   XS             flag previously clear?
557 13616 72           GOYES  SYSFLS         yes: set flag, RTNCC
558 13618 7810         GOSUB  SYSFLC         no: clear flag, RTNSC
559 1361C 02           RTNSC
560            ********************************************************************
561            ********************************************************************
562            **
563            ** Name:    FLGS/C  -  Local flag utilities
564            ** Name:    FLGUP?  -  Local flag utilities
565            **
566            ** Category:   LOCAL
567            **
568            ** Purpose:
569            **      FLGS/C: sets/clears flag according to whether sSTAT=1/0
570            **      FLGUP?: updates flag iff sSTAT=1
571            **
572            ** Entry:
573            **      sSTAT
574            **      D(S) -- new flag value for FLGUP?
575            **      SYSFLS,SYSFLC inputs
576            **
577            ** Exit:
578            **      flag updated as appropriate
579            **
580            ** Calls:      UPDANX
581            **
582            ** Uses.......
583            **  Inclusive: CPU: A(X),B(A),C(A),D0,P
584            **             RAM: ANNAD1-4,FLGREG,SYSFLG
585            **
586            ** Stk lvls:   2
587            **
```

```
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   ----------------------------------
                **  01/05/83      PM         Revised documentation
                **
                ********************************************************************
                ********************************************************************
1361E 870    FLGS/C ?ST=1  =sSTAT
13621 C1            GOYES  SYSFLS
13623 501           GONC   SYSFLC          B.E.T.
13626 860    FLGUP? ?ST=0  =sSTAT
13629 C1            GOYES  D0=D
1362B 1522          A=DAT0 XS
1362F 94F           ?D#0   S
13632 B0            GOYES  SYSFLS
                ********************************************************************
                ********************************************************************
                **
                ** Name:    SYSFLC  -  System flag utilities
                ** Name:    SYSFLS  -  System flag utilities
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Respectively clears or sets bit in A(XS) masked by
                **      C(XS), copies resulting A(XS) nibble into RAM nibble
                **      pointed to by D0, updates annunciators, and restores
                **      PC saved in D(A).
                **
                ** Entry:
                **      A(XS) - contains bit to be cleared or set
                **      C(XS) - mask isolating desired bit
                **      D(A) -- old D0 value
                **      D0 ---- points to RAM destination nibble
                **      HEXMODE
                **
                ** Exit:
                **      flag cleared or set
                **      annunciators updated
                **      D0 -- restored
                **      carry=clear
                **      P=0
                **      HEXMODE
                **
                ** Calls:      UPDANX
                **
                ** Uses.......
                **  Inclusive: CPU: A(X),B(A),C(A),D0,P
                **             RAM: ANNAD1-4,FLGREG,SYSFLG
                **
                ** Stk lvls:   2
                **
                ** History:
                **
```

```
                **    Date     Programmer             Modification
                **  --------   ----------   ---------------------------------
                **  06/23/82      PM         Documented routine
                **  01/14/83      PM         Revised documentation
                **
                ********************************************************************
                ********************************************************************
13634 BAE    SYSFLC C=-C-1 XS              clear flag
13637 0E26          A=A&C  XS
1363B D6            C=A    A              falls through next line unchanged
1363D 0E2E   SYSFLS A=A!C  XS              set flag
13641 782F          GOSUB  UPDANX          update annunciators
13645 DB     DO=D   C=D    A              restore PC
13647 134           DO=C
1364A 03    =rtncc  RTNCC
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) SFLAG?  -  Tests system flag
                **
                ** Category:   GENUTL
                **
                ** Purpose:
                **      Tests a system flag
                **
                ** Entry:
                **      C(B) -- hex flag number (e.g. load FF for -1)
                **      HEXMODE
                **      P=0
                **
                ** Exit:
                **      Carry=Set if flag set
                **      Carry=Clear if flag clear
                **      D(A) - Set to DO
                **      HEXMODE
                **      P=0
                **
                ** Calls:      GTFLAG
                **
                ** Uses.......
                **  Inclusive: A(A),C(15,5-0),D(A)
                **
                ** Stk lvls:   1
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ---------------------------------
                **  06/11/82      PM         Documented routine
                **  04/11/83      PM         Revised documentation
                **
                ********************************************************************
                ********************************************************************
1364C 7E00  =SFLAG? GOSUB  GTFLAG          Test flag
13650 0E26          A=A&C  XS
```

```
698 13654 DB          C=D    A              restore D0
699 13656 134         D0=C
700 13659 BA8         A=-A   XS             carry set/clear as A#0 or =0
701 1365C 01          RTN
702             ****************************************************************
703             ****************************************************************
704             **
705             ** Name:(S) GTFLAG  -  Gets RAM nib and flag mask
706             **
707             ** Category:   GENUTL
708             **
709             ** Purpose:
710             **      Gets nibble and mask for SYSTEM flag specified
711             **      by hex flag #
712             **
713             ** Entry:
714             **      C(B) -- hex flag number
715             **      HEXMODE
716             **      P=0
717             **
718             ** Exit:
719             **      A(XS) - appropriate nibble from flag register
720             **      C(XS) - mask: 1 bit on at position of flag
721             **      D(A) -- previous content of D0
722             **      D0 ---- points at appropriate nibble in flag register
723             **      carry=clear
724             **      P=0
725             **      HEXMODE
726             **
727             ** Calls:      nothing
728             **
729             ** Uses.......
730             **  Inclusive: A(A),C(15,5-0),D(A),D0
731             **
732             ** Stk lvls:   0
733             **
734             ** History:
735             **
736             **    Date      Programmer              Modification
737             **  --------    ----------   ----------------------------------
738             **  06/14/82       PM         Documented routine
739             **  12/17/82       PM         Removed conversion ovfl. tests
740             **  04/11/83       PM         Revised documentation
741             **
742             ****************************************************************
743             ****************************************************************
744 1365E D0   =GTFLAG A=0    A
745 13660 CC          A=A-1  A
746 13662 AEA         A=C    B
747 13665 8A8   NEGFLG ?A=0   A              clear carry for negative flag nbr
748 13668 20          GOYES  POSFLG
749 1366A 3400  POSFLG LC(5)  (=FLGREG)-1    load flag register address
          000
750 13671 136         CD0EX
751 13674 D7          D=C    A
```

```
752 13676 470          GOC    RNGCHK          if argument positive, goto rngchk
753 13679 18F          DO=DO- 16              switch to second bank
754 1367C FC           A=-A-1 A               negate, adjust negative flag nbr
755 1367E D2    RNGCHK C=0    A
756 13680 31F3         LCHEX  3F              load 0003F in C(A)
757 13684 8B6          ?A>C   A               argument out of range?
758 13687 D5           GOYES  ARGOR           if so, goto argerr
759 13689 C4           A=A+A  A
760 1368B D6           C=A    A
761 1368D C6           C=C+C  A
762 1368F 80F1         CPEX   1               (P=0 here)
763 13693 136          CDOEX
764 13696 809          C+P+1
765 13699 136          CDOEX
766 1369C 1522         A=DAT0 XS              read test nibble
767 136A0 C6           C=C+C  A               determine mask load position
768 136A2 C6           C=C+C  A
769 136A4 80D1         P=C    1
770 136A8 0D           P=P-1
771 136AA 3384         LCHEX  1248            load mask (may use C(S))
          21
772 136B0 20           P=     0
773 136B2 03    rtncc2 RTNCC                  return
774             ****************************************************************
775             ****************************************************************
776             **
777             ** Name:    FLGRG?  -  Local flag utilities
778             ** Name:    FLGNB?  -  Local flag utilities
779             **
780             ** Category:   LOCAL
781             **
782             ** Purpose:
783             **      Gets user or system flag info.
784             **
785             ** Entry:
786             **      A(A) --- hex flag number
787             **      Carry=set if user flag
788             **           =clear if system flag
789             **      sSTAT -- negative flag # range test? (FLGRG?)
790             **      HEXMODE
791             **
792             ** Exit:
793             **      GTFLAG exits
794             **
795             ** Calls:      nothing
796             **
797             ** Uses.......
798             **  Inclusive: A(A),C(5-0),D(A),DO
799             **
800             ** Stk lvls:   0
801             **
802             ** History:
803             **
804             **    Date     Programmer              Modification
805             **  --------   ----------   ------------------------------------
```

```
806             **  01/05/83      PM         Documented routine
807             **
808             ********************************************************************
809             ********************************************************************
810 136B4 45B   FLGRG? GOC    POSFLG
811 136B7 860         ?ST=0  =sSTAT
812 136BA BA          GOYES  NEGFLG
813 136BC 4DA   FLGNB? GOC    POSFLG
814                                     test for neg flag nbr in range
815 136BF 31FD        LCHEX  DF          load (set/clear bound) - 1
816 136C3 9E6         ?A>C   B           flag nbr within range?
817 136C6 F9          GOYES  NEGFLG        yes: get flag
818 136C8 5B1         GONC   ARGOR         no: arg-out-of-range
819             ********************************************************************
820             ********************************************************************
821             **
822             ** Name:(S) RNDAHX  -  Pops, tests,rounds,converts dec to hex
823             **
824             ** Category:   MTHUTL
825             **
826             ** Purpose:
827             **      Pops, tests, rounds, and converts a real number
828             **      to hex integer.
829             **
830             ** Entry:
831             **      number to be rounded and converted on top of
832             **        math stack
833             **
834             ** Exit:
835             **      A(A) -- rounded hex integer
836             **      Carry=Clear: negative integer
837             **      Carry=Set:   nonnegative integer (incl -0)
838             **      fatal error if array or complex type, or NaN
839             **      HEXMODE
840             **      XM=0
841             **      P=0
842             **
843             ** Calls:      ARGST-,DCHXF
844             **
845             ** Uses.......
846             **  Inclusive: A,B(S,A),C(A),D(A),P,SB,XM  unless fatal error
847             **
848             ** Stk lvls:   3
849             **
850             ** NOTE:
851             **           Input              Fatal Error Message
852             **
853             **           array                  "eDATTY"
854             **         complex                  "eDATTY"
855             **           NaN                    "eIVARG"
856             **    conversion overflow           "eIVARG"
857             **
858             ** History:
859             **
860             **    Date     Programmer             Modification
```

```
861            ** --------   ----------   ----------------------------------
862            ** 06/11/82      PM         Documented routine
863            ** 08/11/82      PM         Redefined fatal error exits
864            ** 12/17/82      PM         Fatal error for convers. ovfl.
865            ** 02/25/83      PM         Removed unnecessary GOC
866            **
867            ********************************************************************
868            ********************************************************************
869 136CB 8E00 =RNDAHX GOSUBL =ARGST-        pop, test argument
       00
870 136D1 8E00        GOSUBL =DCHXF         rounds, converts to hex integer
       00
871 136D7 400         RTNC                  return with carry set if positive
872 136DA 8A8         ?A=0   A              treat 0 as positive
873 136DD 00          RTNYES
874 136DF 831         ?XM=0                 negative and in-range?
875 136E2 0D          GOYES  rtncc2           yes: return with carry clear
876 136E4 8C00  ARGOR GOLONG =IVAERR          no:  invalid argument error
       00
877            ********************************************************************
878 136EA 8C00 =expex- GOLONG =EXPEX-        collapses math stack
       00
879            ********************************************************************
880            ********************************************************************
881            **
882            ** Name:    TRAP    -  User's Trap Function
883            **
884            ** Category:   FNEXEC
885            **
886            ** Purpose:
887            **      Returns the value of the trap specified by the
888            **      IEEE exception flag number.  If an optional trap
889            **      value is included, sets trap to that value and
890            **      returns the previous value.
891            **
892            ** Entry:
893            **      argument(s) on top of math stack
894            **      C(S) -- argument count
895            **      HEXMODE
896            **
897            ** Exit:
898            **      value on top of math stack
899            **
900            ** Calls:      PARMCT,RNDAHX
901            **             exits through FNRTN4
902            **
903            ** Uses.......
904            **  Inclusive: CPU: A,B(S,A),C,D(A),P,R0,D1,sSTAT,SB,XM
905            **             RAM: TRPREG
906            **             unless fatal error
907            **
908            ** Stk lvls:   4
909            **
910            ** History:
911            **
```

```
912                ** Date     Programmer          Modification
913                ** -------- ----------  ---------------------------------
914                ** 06/11/82     PM       Documented routine
915                ** 02/25/83     PM       Added conditional jumps for sign
916                **
917                ********************************************************************
918                ********************************************************************
919 136F0 8812             NIBHEX 8812
920 136F4 746E =TRAP       GOSUB  PARMCT          set sSTAT according to # parms
921 136F8 461              GOC    NOTVAL
922 136FB 7CCF             GOSUB  RNDAHX          pop, test, round and convert
923 136FF 54E              GONC   ARGOR           error if negative or out-of-range
924 13702 17F              D1=D1+ 16                trap value to hex
925 13705 100              R0=A                   R0(A):=new trap value
926 13708 F4               ASR    A
927 1370A 8AC              ?A#0   A
928 1370D 7D               GOYES  ARGOR           error if value <0 or >F
929 1370F 78BF  NOTVAL     GOSUB  RNDAHX          pop, test, round and convert to hex
930 13713 40D              GOC    ARGOR           error if positive (have FFFFF for
931 13716 D2               C=0    A                    out-of-range)
932 13718 CE               C=C-1  A
933 1371A 3100             LC(2)  =flIVL          load IVL flag number
934 1371E EA               A=A-C  A
935 13720 43C              GOC    ARGOR           error if flag # <IVL or >=0
936 13723 304              LCHEX  4               number of exceptions - 1
937 13726 986              ?A>C   P
938 13729 BB               GOYES  ARGOR           error if flag # >INX
939 1372B 3400             LC(5)  (=TRPREG)+4
          000
940 13732 E2               C=C-A  A
941 13734 136              CDOEX                  D0 points to trap nibble
942 13737 D7               D=C    A               D(A) has previous D0 value
943 13739 AF0              A=0    W
944 1373C 2E               P=     14
945 1373E 1520             A=DAT0 P
946 13742 30A              LCHEX  A
947 13745 982              ?A<C   P               value<A?
948 13748 E0               GOYES  TRAP1
949 1374A B0A              A=A-C  P
950 1374D BD4              ASR    M
951 13750 B04              A=A+1  P
952 13753 B64              A=A+1  B               * increment exponent
953 13756 860   TRAP1      ?ST=0  =sSTAT
954 13759 90               GOYES  TRPEND
955 1375B 118              C=R0                   put new trap value
956 1375E 15C0             DAT0=C 1
957 13762 DB    TRPEND     C=D    A               restore D0
958 13764 134              D0=C
959 13767 AF6              C=A    W
960 1376A 8C00             GOLONG =FNRTN4
          00
961                ********************************************************************
962                ********************************************************************
963                **
964                ** Name:    DEFALT  -  Default statement
```

```
 965            **
 966            ** Category:   STEXEC
 967            **
 968            ** Purpose:
 969            **      Sets arithmetic trap values according to a keyword:
 970            **      ON, OFF, or EXTEND.
 971            **
 972            ** Entry:
 973            **      D0 -- program counter, points to keyword
 974            **      P=0
 975            **      HEXMODE
 976            **
 977            ** Exit:
 978            **      trap values set
 979            **      P=0
 980            **      HEXMODE
 981            **
 982            ** Calls:      nothing
 983            **             exits through NXTSTM
 984            **
 985            ** Uses.......
 986            **  Inclusive: CPU: A(B),B(B),C(A),D0
 987            **             RAM: TRPREG
 988            **
 989            ** Stk lvls:   0
 990            **
 991            ** Detail:
 992            **      keyword      trap values (INX,UNF,OVF,DVZ,IVL)
 993            **
 994            **        OFF           10000
 995            **        ON            11111
 996            **        EXTEND        22221
 997            **
 998            **
 999            ** History:
1000            **
1001            **    Date     Programmer              Modification
1002            **   --------   ----------   ----------------------------------
1003            **   06/11/82      PM        Documented routine
1004            **   07/30/82      PM        Reversed meaning of trap values 0,2
1005            **   01/05/83      PM        Revised documentation
1006            **
1007            ***************************************************************
1008            ***************************************************************
1009 13770 0000         REL(5) =DEFADC
         0
1010 13775 0000         REL(5) =DEFAP
         0
1011 1377A 14A  =DEFALT A=DAT0 B              get token
1012 1377D 3100         LC(2)  =tON           load ON token
1013 13781 AE5          B=C    B              *
1014 13784 3411         LCHEX  11111          trap values for ON
         111
1015 1378B 960          ?A=B   B
1016 1378E 81           GOYES  DEFOUT
```

```
1017 13790 3100          LC(2)  =tOFF          load OFF token
1018 13794 AE5           B=C    B              *
1019 13797 3322          LCHEX  2222           trap values for EXTEND
           22
1020 1379D 964           ?A#B   B              not OFF?
1021 137A0 60            GOYES  DEFOUT
1022 137A2 D2            C=0    A              trap values for OFF
1023 137A4 E6            C=C+1  A
1024 137A6 1B00  DEFOUT  D0=(5) =TRPREG
           000
1025 137AD 144           DAT0=C A
1026 137B0 652D          GOTO   nxtstm
1027               ****************************************************************
1028               ****************************************************************
1029               **
1030               ** Name:    OPTION  -  Select base, angle, or round mode
1031               **
1032               ** Category:   STEXEC
1033               **
1034               ** Purpose:
1035               **      Sets BASE, ANGLE, or ROUND mode
1036               **
1037               ** Entry:
1038               **      D0 -- points to OPTION keyword token
1039               **      P=0
1040               **      HEXMODE
1041               **
1042               ** Exit:
1043               **      BASE, ANGLE, or ROUND mode set
1044               **      HEXMODE
1045               **
1046               ** Calls:      expex-,RNDAHX,SFLAG*,SFLG*B
1047               **             exits through NXTSTM
1048               **
1049               ** Uses.......
1050               **  Exclusive of EXPEX-: CPU: A,B(A),C,D(A),P,D0,SB,XM
1051               **                       RAM: ANNAD1,SYSFLG
1052               **
1053               ** Stk lvls:   BASE:        max(4,1+EXPEXC)
1054               **             ANGLE,ROUND: 3
1055               **
1056               ** History:
1057               **
1058               **    Date     Programmer             Modification
1059               **  --------   ----------   ---------------------------------
1060               **  06/14/82      PM         Documented routine
1061               **  10/19/82      PM         Added ROUND mode
1062               **  01/05/83      PM         Revised documentation
1063               **
1064               ****************************************************************
1065               ****************************************************************
1066 137B4 0000          REL(5) =OPTDC
           0
1067 137B9 0000          REL(5) =OPTP
           0
```

```
1068 137BE 1527 =OPTION A=DAT0 W            Option statement execution
1069 137C2 AF6          C=A    W
1070 137C5 3100         LC(2)  =tBASE
1071 137C9 962          ?A=C   B             BASE?
1072 137CC 07           GOYES  OPBAS         If so, goto OPBAS
1073 137CE 3500         LC(6)  =tANGLE
           0000
1074 137D6 972          ?A=C   W             ANGLE?
1075 137D9 05           GOYES  OPANGL        If so, goto OPANGL
1076 137DB 165          DO=DO+ 6             ROUND mode: skip over token
1077 137DE 15A5         A=DAT0 6
1078 137E2 3100         LC(2)  =flINFR
1079 137E6 D5           B=C    A
1080 137E8 3500         LC(6)  =tPOS
           0000
1081 137F0 972          ?A=C   W
1082 137F3 92           GOYES  OPINFR
1083 137F5 3500         LC(6)  =tZERO
           0000
1084 137FD 972          ?A=C   W
1085 13800 11           GOYES  OZEROR
1086 13802 3500         LC(6)  =tNEG
           0000
1087 1380A 976          ?A#C   W
1088 1380D 11           GOYES  ONEARR
1089 1380F 21           P=     1             round to -inf
1090 13811 7CDD  OZEROR GOSUB  SFLG*B        round to zero
1091 13815 3100         LC(2)  =flNEGR
1092 13819 5F5          GONC   snxtst        B.E.T.
1093               *
1094 1381C 21    OPINFR P=     1             round to +inf
1095 1381E 7FCD  ONEARR GOSUB  SFLG*B        round to nearest
1096 13822 3100         LC(2)  =flNEGR
1097 13826 545          GONC   cnxtst        B.E.T.
1098 13829 165   OPANGL DO=DO+ 6             ANGLE mode: skip over token
1099 1382C 15A1         A=DAT0 2             read next token
1100 13830 3100         LC(2)  =tDEGRE
1101 13834 962          ?A=C   B             DEGREE?
1102 13837 C2           GOYES  DEGREE
1103 13839 5B3          GONC   RADIAN
1104 1383C 161   OPBAS  DO=DO+ 2             BASE mode: skip over token
1105 1383F 77AE         GOSUB  expex-        Evaluate expression
1106 13843 748E         GOSUB  RNDAHX        pop, test, round, convert base
1107 13847 3100         LC(2)  =flBASE       load BASE flag #
1108 1384B CC           A=A-1  A             base=0?
1109 1384D 4D2          GOC    cnxtst          yes: clear BASE flag
1110 13850 CC           A=A-1  A             base=1?
1111 13852 462          GOC    snxtst          yes: set BASE flag
1112 13855 6E8E         GOTO   ARGOR         base#0 or 1: fatal error
1113               ****************************************************************
1114               ****************************************************************
1115               **
1116               ** Name:    DEGREE  -  Sets degrees mode
1117               **
1118               ** Category:   STEXEC
```

```
1119            **
1120            ** Purpose:
1121            **      Sets ANGLE mode to DEGREES (short statement form of
1122            **      OPTION ANGLE DEGREES)
1123            **
1124            ** Entry:
1125            **      P=0
1126            **      HEXMODE
1127            **
1128            ** Exit:
1129            **      ANGLE mode set to DEGREES
1130            **
1131            ** Calls:       SFLAG*
1132            **              exits through NXTSTM
1133            **
1134            ** Uses.......
1135            **  Inclusive: CPU: A(A),B(A),C(5-0),D(A),P,D0
1136            **              RAM: ANNAD1,SYSFLG
1137            **
1138            ** Stk lvls:   3
1139            **
1140            ** History:
1141            **
1142            **    Date      Programmer              Modification
1143            **  --------    ----------   ----------------------------------
1144            **  06/14/82      PM         Documented routine
1145            **  01/05/83      PM         Revised Documention
1146            **
1147            ********************************************************************
1148            ********************************************************************
1149 13859 0000          REL(5) =STOPDC
         0
1150 1385E 0000          REL(5) =RTNCC
         0
1151 13863 3100 =DEGREE LC(2)  =flRAD          select RAD flag
1152 13867 6310         GOTO   cnxtst          clear it
1153            ********************************************************************
1154            ********************************************************************
1155            **
1156            ** Name:    RADIAN  -  Sets radians mode
1157            **
1158            ** Category:   STEXEC
1159            **
1160            ** Purpose:
1161            **      Sets ANGLE mode to RADIANS (short statement form of
1162            **      OPTION ANGLE RADIANS)
1163            **
1164            ** Entry:
1165            **      P=0
1166            **      HEXMODE
1167            **
1168            ** Exit:
1169            **      ANGLE mode set
1170            **
1171            ** Calls:       SFLAG*
```

```
1172            **              exits through NXTSTM
1173            **
1174            ** Uses.......
1175            **  Inclusive: CPU: A(A),B(A),C(5-0),D(A),D0,P
1176            **             RAM: ANNAD1,SYSFLG
1177            **
1178            ** Stk lvls:   3
1179            **
1180            ** History:
1181            **
1182            **    Date     Programmer              Modification
1183            **  --------   ----------   ------------------------------------
1184            **  06/14/82      PM         Documented routine
1185            **  01/05/83      PM         Revised documentation
1186            **
1187            ********************************************************************
1188            ********************************************************************
1189 13868 0000         REL(5) =STOPDC
         0
1190 13870 0000         REL(5) =RTNCC
         0
1191 13875 3100 =RADIAN LC(2)  =flRAD          select RAD flag
1192 13879 21    snxtst P=     1               set flag
1193 1387B 747D  cnxtst GOSUB  SFLAG*          clear flag
1194 1387F 665C         GOTO   nxtstm          Jump to next statement
1195            ********************************************************************
1196            ********************************************************************
1197            **
1198            ** Name:     INX     -  Flag Name Functions
1199            ** Name:     UNF     -  Flag Name Functions
1200            ** Name:     OVF     -  Flag Name Functions
1201            ** Name:     DVZ     -  Flag Name Functions
1202            ** Name:     IVL     -  Flag Name Functions
1203            **
1204            ** Category:   FNEXEC
1205            **
1206            ** Purpose:
1207            **      Return, respectively, the inexact, underflow, overflow,
1208            **      divide-by-zero, and invalid operation system flag
1209            **      numbers.
1210            **
1211            ** Entry:
1212            **      Carry=clear
1213            **
1214            ** Exit:
1215            **      flag number pushed onto math stack
1216            **
1217            ** Calls:      exits through FNRTN1
1218            **
1219            ** Uses.......
1220            **  Inclusive: C,P
1221            **
1222            ** Stk lvls:   0
1223            **
1224            ** History:
```

```
1225            **
1226            **    Date      Programmer             Modification
1227            **  --------    ----------   ----------------------------------
1228            **  06/10/82       PM         Documented routine
1229            **  01/05/83       PM         Revised documentation
1230            **
1231            ********************************************************************
1232            ********************************************************************
1233 13883 00           NIBHEX 00
1234 13885 24   =INX    P=     4               inexact result (-4)
1235 13887 5B1          GONC   LOADNB          B.E.T.
1236 1388A 00           NIBHEX 00
1237 1388C 25   =UNF    P=     5               underflow (-5)
1238 1388E 541          GONC   LOADNB          B.E.T.
1239 13891 00           NIBHEX 00
1240 13893 26   =OVF    P=     6               overflow (-6)
1241 13895 5D0          GONC   LOADNB          B.E.T.
1242 13898 00           NIBHEX 00
1243 1389A 27   =DVZ    P=     7               divide by zero (-7)
1244 1389C 560          GONC   LOADNB          B.E.T.
1245 1389F 00           NIBHEX 00
1246 138A1 28   =IVL    P=     8               invalid operation (-8)
1247 138A3 AF2   LOADNB C=0    W
1248 138A6 05           SETDEC
1249 138A8 BCE          C=-C-1 S
1250 138AB 80FE         CPEX   14
1251 138AF 8C00         GOLONG =FNRTN1
             00
1252 138B5              END
```

```
 ANNAD1  Ext                    -   369
 ANNAD2  Ext                    -   355
 ANNAD3  Ext                    -   361
 ANNAD4  Ext                    -   351
 ARGOR   Abs   79588 #136E4     -   876   758   818   923   928   930   935   938
                                   1112
 ARGST-  Ext                    -   869
 BABBAJ  Ext                    -   263
=CFLAG   Abs   78994 #13492     -    92
 CFLAG2  Abs   79019 #134AB     -    99    95
 CNFLCT  Ext                    -   264
 DO=D    Abs   79429 #13645     -   655   600
 DBLUP   Abs   79314 #135D2     -   414   352   362
 DCHXF   Ext                    -   870
 DEFADC  Ext                    -  1009
=DEFALT  Abs   79738 #1377A     -  1011
 DEFAP   Ext                    -  1010
 DEFOUT  Abs   79782 #137A6     -  1024  1016  1021
=DEGREE  Abs   79971 #13863     -  1151  1102
=DVZ     Abs   80026 #1389A     -  1243
 EXPEX-  Ext                    -   878
=FLAG    Abs   79127 #13517     -   246
 FLALL   Abs   79066 #134DA     -   150    61    92
 FLGNB?  Abs   79548 #136BC     -   813   102
 FLGREG  Ext                    -   154   346   364   749
 FLGRG?  Abs   79540 #136B4     -   810   257
 FLGS/C  Abs   79390 #1361E     -   596   103
 FLGUP?  Abs   79398 #13626     -   599   261
 FLMATH  Abs   79090 #134F2     -   196    68    94
 FNRTN1  Ext                    -  1251
 FNRTN4  Ext                    -   960
 FORSTK  Ext                    -   107
=GTFLAG  Abs   79454 #1365E     -   744   468   510   553   696
=INX     Abs   80005 #13885     -  1234
 IVAERR  Ext                    -   876
=IVL     Abs   80033 #138A1     -  1246
 LOADNB  Abs   80035 #138A3     -  1247  1235  1238  1241  1244
 NEGFLG  Abs   79461 #13665     -   747   812   817
 NOFVAL  Abs   79160 #13538     -   256   247   254
 NOTVAL  Abs   79631 #1370F     -   929   921
 ONEARR  Abs   79902 #1381E     -  1095  1088
 OPANGL  Abs   79913 #13829     -  1098  1075
 OPBAS   Abs   79932 #1383C     -  1104  1072
 OPINFR  Abs   79900 #1381C     -  1094  1082
 OPTDC   Ext                    -  1066
=OPTION  Abs   79806 #137BE     -  1068
 OPTP    Ext                    -  1067
=OVF     Abs   80019 #13893     -  1240
 OZEROR  Abs   79889 #13811     -  1090  1085
 PARMCT  Abs   79196 #1355C     -   298   246   920
 POP1N+  Ext                    -   248
 POSFLG  Abs   79466 #1366A     -   749   748   810   813
=RADIAN  Abs   79989 #13875     -  1191  1103
=RNDAHX  Abs   79563 #136CB     -   869   101   256   922   929  1106
 RNGCHK  Abs   79486 #1367E     -   755   752
```

```
 RTNCC   Ext                  -  1150  1190
=SFLAG   Abs   78931 #13453  -    61
=SFLAG*  Abs   79347 #135F3  -   465  1193
 SFLAG0  Abs   78955 #1346B  -    68    62
 SFLAG2  Abs   78973 #1347D  -    73    69
=SFLAG?  Abs   79436 #1364C  -   696
=SFLAGC  Abs   79361 #13601  -   510   466
=SFLAGS  Abs   79354 #135FA  -   468
=SFLAGT  Abs   79368 #13608  -   553
 SFLG*B  Abs   79345 #135F1  -   464  1090  1095
 SFLGDC  Ext                  -    59    90
 SFLGP   Ext                  -    60    91
 SNGLP+  Abs   79318 #135D6  -   416   370
 SNGLUP  Abs   79316 #135D4  -   415   356
 STMEND  Ext                  -   111
 STOPDC  Ext                  -  1149  1189
 SYSFLC  Abs   79412 #13634  -   650   511   558   598
 SYSFLG  Ext                  -   201
 SYSFLS  Abs   79421 #1363D  -   653   469   557   597   603
=TRAP    Abs   79604 #136F4  -   920
 TRAP1   Abs   79702 #13756  -   953   948
 TRPEND  Abs   79714 #13762  -   957   954
 TRPREG  Ext                  -   939  1024
=UNF     Abs   80012 #1388C  -  1237
 UPDAN1  Abs   79304 #135C8  -   369   350
=UPDANN  Abs   79217 #13571  -   345    66
=UPDANX  Abs   79213 #1356D  -   344   654
 XFLAG3  Abs   79026 #134B2  -   101    75   110
 anxtsB  Abs   78945 #13461  -    65    72    98
 anxtsW  Abs   78941 #1345D  -    64    93
 cnflct  Abs   79190 #13556  -   264   249
 cnxtst  Abs   79995 #1387B  -  1193  1097  1109  1152
=expex-  Abs   79594 #136EA  -   878    73    99  1105
 flBASE  Ext                  -  1107
 flINFR  Ext                  -  1078
 flIVL   Ext                  -   933
 flNEGR  Ext                  -  1091  1096
 flRAD   Ext                  -  1151  1191
 nxtstm  Abs   79062 #134D6  -   111    67  1026  1194
=rtncc   Abs   79434 #1364A  -   657
 rtncc2  Abs   79538 #136B2  -   773   875
 sSTAT   Ext                  -    74   100   298   302   596   599   811   953
 snxtst  Abs   79993 #13879  -  1192  1092  1111
 tALL    Ext                  -   151
 tANGLE  Ext                  -  1073
 tBASE   Ext                  -  1070
 tDEGRE  Ext                  -  1100
 tMATH   Ext                  -   197
 tNEG    Ext                  -  1086
 tOFF    Ext                  -  1017
 tON     Ext                  -  1012
 tPOS    Ext                  -  1080
 tZERO   Ext                  -  1083
```

Input Parameters

  Source file name is PM&FLG::MS

  Listing file name is PM/FLG:TI:ML::-1

  Object file name is PM%FLG:TI:MS::-1

```
                                  111111
                        0123456789012345
```
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS    CCC    &     DDDD     A    TTTTT
 2            *    S   S  C   C  & &     D  D   A A     T
 3            *    S      C      & &     D  D  A   A    T
 4            *     SSS   C       &      D  D  A   A    T
 5            *        S  C      & & &   D  D  AAAAA    T
 6            *    S   S  C   C  &  &    D  D  A   A    T
 7            *     SSS    CCC    && &  DDDD   A   A    T
 8            *
 9            *
10
11            *
12                     TITLE  Store & Retrieve Data - PRINT#/READ#<831216.1958>
13 138B5               ABS    #138B5
14                     RDSYMB TI%EQU::MS
15                     RDSYMB SB%RAM::MS
16
17            ****************************************************************
18            *
19            *              Local Status Symbol Definitions
20            *
21            ****************************************************************
22             Notnum EQU    0    Data item on math stack is not a number.
23             Array  EQU    1    Data item on math stack is an array vector.
24             String EQU    2         //                 is a string.
25             Cmplex EQU    3         //                 is a complex number.
26             Return EQU    4    Do a return after expanding the file.
27             Error  EQU    5    Error had been happen.
28             Odd    EQU    6    The string length in the TEXT file is odd.
29             Done   EQU    6    Done with entire string.
30             Check  EQU    7    Only check the array vector on stack
31             UpMstk EQU    7    UPdate math stack pointer.
32             Fresh  EQU    8    Doing on the 1st element of an array.
33             Random EQU    9    Random access.
34             I/Obuf EQU   10    The file is in a mass memory device.
35             Iram   EQU   11    The file is in Independent RAM.
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) WSTRFX  -  Write a String to a DATA File
                **
                ** Category:   FILUTL
                **
                ** Purpose:  Write a string to the fixed length data file
                **           If the file is in an external mass memory device,
                **           data will be written to its I/O buffer first. When
                **           the I/O buffer is full or the file is closed, the
                **           content of the I/O buffer will be written back to
                **           the file.
                **
                ** Entry:  A = string length in bytes
                **         B = # of bytes left in current record
                **         RO(A)= Current file pointer
                **         RO(15,14)=Current byte ptr in file I/O buffer
                **         R1= record length in bytes
                **         D1 @ past the string ( String is stored backward)
                **         STMTD1 Contains FIB entry address
                **         S9 = 0 if serial access
                **            = 1 if random access
                **         S10 = 0 if internal file
                **             = 1 if external file(write file I/O buffer)
                **         It is assumed that there is enough room left in the
                **           file.
                **         If the string is too long to fit into the current
                **          record and it is a serial access, the string will be
                **          broken down into smaller logical units.
                **
                ** Exit:  Carry set => Random access crossing record boundary
                **        Carry clear => Done successfully
                **          RO(15,14) & RO(A) will be maintained
                **
                ** Calls:  DO+2WR, WRBYTC
                **
                ** Uses:   A,B,C,DO,RO, ST[4-0]
                **
                ** Stk lvls:  1 if internal file
                **            4 if external file (when flush file buffer)
                ****************************************************************
                ****************************************************************
                *
138B5 118       =WSTRFX C=RO
138B8 134               DO=C                    DO @ CURRENT FILE POINTER
138BB D2                C=0     A
138BD 303               LC(1)   3
138C0 8B9               ?B>=C   A               CAN HEADER FIT IN CURRENT RECORD?
138C3 E1                GOYES   WSTF10          IF SO, GOTO WSTR10
138C5 879               ?ST=1   Random
138C8 00                RTNYES
138CA CD        WSTF05  B=B-1   A
138CC 4A0               GOC     WSTF08
```

```
 91 138CF 73A1          GOSUB  DO+2WR
 92 138D3 66FF          GOTO   WSTF05
 93             *
 94 138D7 119   WSTF08  C=R1
 95 138DA D5            B=C    A              B= RECORD LENGTH IN BYTES
 96 138DC D2            C=0    A
 97 138DE 303           LC(1)  3
 98 138E1 E1    WSTF10  B=B-C  A              B=BYTES AVAILABLE IN CURRENT REC.
 99 138E3 846           ST=0   Done
100 138E6 31FD          LC(2)  =fAOS          ASSUME FIT CURRENT RECORD
101 138EA 8BC           ?B>=A  A              CAN WHOLE STRING FIT CURRENT REC.?
102 138ED B0            GOYES  WSTF30         IF SO, GOTO WRITE STRING LENGTH
103 138EF 879           ?ST=1  Random         RANDOM ACCESS ?
104 138F2 00            RTNYES                IF SO, RETURN WITH CARRY SET
105 138F4 31FC          LC(2)  =fSOS          HEADER SAY START OF STRING ONLY
106 138F8 7771  WSTF30  GOSUB  WRBYTC         WRITE FIRST BYTE OF STRING HEADER
107 138FC AE6           C=A    B
108 138FF 7071          GOSUB  WRBYTC         WRITE 1ST BYTE OF STRING LENGTH
109 13903 D6            C=A    A
110 13905 F6            CSR    A
111 13907 BB6           CSR    X              C(B)= 2ND BYTE OF STRING LENGTH
112 1390A 7561          GOSUB  WRBYTC
113 1390E E0            A=A-B  A              A= REMAIN STRING LENGTH + 3
114 13910 590           GONC   WSTF40
115 13913 C0            A=A+B  A
116 13915 D8            B=A    A
117 13917 856           ST=1   Done
118 1391A CD    WSTF40  B=B-1  A              DECREMENT BYTE COUNTER
119 1391C 401           GOC    WSTF50         IF CARRY, DONE WITH ONE RECORD
120 1391F 1C1           D1=D1- 2
121 13922 14F           C=DAT1 B
122 13925 7A41          GOSUB  WRBYTC
123 13929 60FF          GOTO   WSTF40         (B.E.T.)
124 1392D 8A8   WSTF50  ?A=0   A              ALL DONE ?
125 13930 32            GOYES  WSTF70         YES, JUST FIT INTO ONE RECORD
126 13932 876           ?ST=1  Done
127 13935 E1            GOYES  WSTF70
128 13937 119           C=R1
129 1393A D5            B=C    A
130 1393C D2            C=0    A
131 1393E 303           LC(1)  3
132 13941 E1            B=B-C  A              C= BYTES AVAILABLE IN CURRENT REC.
133 13943 31F6          LC(2)  =fEOS          END OF STRING HEADER
134 13947 8BC           ?A<=B  A              REMAIN STRING FIT IN ONE RECORD ?
135 1394A EA            GOYES  WSTF30         IF SO, WRITE END OF STRING HEADER
136 1394C 31F7          LC(2)  =fMOS          MIDDLE OF STRING HEADER
137 13950 57A           GONC   WSTF30         (B.E.T.)
138            ****************************************************************
139            **
140            ** Bug fixing 8/4/83 after TI%RM5
141            ** In random access PRINT #, if the last string in a record fill
142            ** up to the last byte of the record, an end of record mark will
143            ** still be written to the beginning of the next record.
144            **
145            ****************************************************************
```

```
146             *
147             *WSTF70 GOSUB  UPCPOS
148             *       ?ST=0  I/Obuf          Old code looks like this block
149             *       GOYES  WSTF80
150             *       ?D=0   A
151             *       GOYES  WSTF90
152             ****************************************************************
153             *
154 13953 7013  WSTF70 GOSUB  UPCPOS
155 13957 451          GOC    WSTF90          No EOR if random & end of record
156 1395A 8AF          ?D#0   A               Reached end of file yet ?
157 1395D 40           GOYES  WSTF80          If not, write EOR
158 1395F 03           RTNCC                 ***Pack here...could save 2 nibs**
159 13961 31FE  WSTF80 LC(2)  =fEOR
160 13965 7A01         GOSUB  WRBYTC
161 13969 7F91         GOSUB  BACK1B
162 1396D 03    WSTF90 RTNCC
163             *
164             ****************************************************************
165             ****************************************************************
166             **
167             ** Name:(S) WRTSTR  -  Write a string to an open TEXT file
168             **
169             ** Category:   FILUTL
170             **
171             ** Purpose:
172             **      Write a string on stack to an open TEXT file.
173             **
174             **      The string will be written out as:
175             **      +--------+----------------------------+----+
176             **      | Length |     String                 | Pad|
177             **      +--------+----------------------------+----+
178             **       2 bytes        n bytes                  1 bytes
179             **
180             **      The pad is not included in the length and it will be
181             **      there only if the string length is an odd number.
182             **
183             ** Entry:
184             **     D1 @ string length(2 nibs past the string header on
185             **          math stack).
186             **     R0 = Current file pointer
187             **     S6 =1 If length odd
188             **     STMTD1 = Entry address in FIB
189             **     It is assumed that there is enough room left in the
190             **        file to store the string.
191             **
192             ** Exit:
193             **    R0(A) will be updated
194             **    The string is popped and the AVMEME is update.
195             **    Current position in FIB will be updated too.
196             **
197             ** Calls:      DROPST,WRBYTC,UPCPOS,WRTEOF,BACK2B,SETWRT
198             **
199             ** Uses.......
200             **  Inclusive: A,B,C,D0,D1,ST[4-0]     ???
```

```
                **
                ** Stk lvls:
                **              Internal file:  2
                **              External file:  4
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ---------------------------------
                **              SC            Wrote
                **  11/05/83    BS            Updated documentation
                **
                ********************************************************************
                ********************************************************************
                *
13960F 7A54 =WRTSTR GOSUB  DROPST           SET D1 PTS TO END OF STRING
13973 131   =WRSTR* D1=A                   STRING IS WRITTEN BACKWARD
13976 81E           CSRB                    C= LENGTH IN BYTES
13979 D7            D=C    A                D= STRING LENGTH IN BYTES
1397B 110           A=R0
1397E 130           D0=A                    D0 @ CURRENT FILE POINTER
13981 DA            A=C    A
13983 F6            CSR    A
13985 BB6           CSR    X
13988 77E0          GOSUB  WRBYTC           WRITE MS BYTE OF STRING LENGTH
1398C D6            C=A    A
1398E 71E0          GOSUB  WRBYTC           WRITE LS BYTE OF STRING LENGTH
13992 1C1    WSTR20 D1=D1- 2
13995 14F           C=DAT1 B
13998 CC            A=A-1  A
1399A 4A0           GOC    WSTR30           ALL DONE
1399D 72D0          GOSUB  WRBYTC
139A1 60FF          GOTO   WSTR20           (B.E.T.)
139A5 866    WSTR30 ?ST=0  Odd
139A8 60            GOYES  WSTR40           LENGTH EVEN
139AA 75C0          GOSUB  WRBYTC           PUT IN PAD
139AE 75B2   WSTR40 GOSUB  UPCPOS
139B2 75A0          GOSUB  WRTEOF           WRITE EOF MARK
139B6 71A0          GOSUB  WRTEOF
139BA 7C41          GOSUB  BACK2B           SEE IF CROSS END OF BUFFER
139BE 7631          GOSUB  SETWRT           SET ACCESS MODE TO WRITE
139C2 03            RTNCC                   **Pack here...could save 2 nibs***
                *
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) WRTNUM  -  Write a Number to DATA or SDATA file.
                **
                ** Category:   FILUTL
                **
                ** Purpose:  Write a number from math stack to a file of type
                **           DATA or SDATA.
                **
                **  A number will always be written out as a real(8 bytes):
                **
```

```
256                ** High                                                Low
257                **  ------------------------------------------------------
258                **  | M0,M1| M2,M3| M4,M5| M6,M7| M8,M9|M10,M11| E0,MS| E1,E2|
259                **  ------------------------------------------------------
260                **
261                ** Entry:  A= the number (internal form)
262                **         S10 = 0 if the file is in memory.
263                **             = 1 if the file is in an ext. mass memory device.
264                **         D0 @ Current file pointer
265                **           If the file is in memory, D0 is directly pointing
266                **           at the file.
267                **           If the file is in an external mass memory device,
268                **           D0 is pointing at the I/O buffer of the file and
269                **           R0(15,14) = Byte pointer of the file I/O buffer
270                **         D1 @ Top of stack
271                ** Exit:   D1 will drop 16(D1=D1-16) and stored to MTHSTK
272                **         D0 Past the number
273                **
274                ** Used:  A,B,C,D0,D1
275                **
276                ** Detail: The number will be formatted and written on the math
277                **         stack first, and then it will be written out to the
278                **         file or I/O buffer one byte at a time. If is written
279                **         to an I/O buffer, when the buffer gets full, this
280                **         routine will POLL the HP-IL ROM to dump the buffer to
281                **         the device and read in the next buffer.
282                **
283                ********************************************************************
284                ********************************************************************
285                *
286 139C4 149   =WRTNUM DAT1=A B              WRITE THE NUMBER BACK TO STACK 1ST
287 139C7 171          D1=D1+ 2
288 139CA 1514         DAT1=A S
289 139CE 170          D1=D1+ 1
290 139D1 1512         DAT1=A XS
291 139D5 170          D1=D1+ 1
292 139D8 1515         DAT1=A M
293 139DC 1C3          D1=D1- 4
294                *
295 139DF 307          LC(1)  7
296 139E2 816          CSRC
297 139E5 ACA          A=C    S              A(S) = 7, LOOP COUNTER
298 139E8 7180 WRTN10  GOSUB  WRBYTD
299 139EC A4C          A=A-1  S
300 139EF 58F          GONC   WRTN10
301 139F2 7FF3         GOSUB  UPMTHS
302 139F6 6072         GOTO   UPCPOS
303                *
304                *
305                ********************************************************************
306                ********************************************************************
307                **
308                ** Name:    SKPBYT  -  Skip a Number of Bytes in an Opened File.
309                **
310                ** Category:   FILUTL
```

```
311             **
312             ** Purpose:  Skip a given number of bytes in an opened file
313             **           The file can be in RAM/ROM or in an external device
314             **
315             ** Entry:  D0 @ current position of the file.
316             **          (Absolute addr if file in RAM/ROM)
317             **          (Absolute addr @ the file I/O buffer for ext. file)
318             **         A = # of bytes to skip
319             **         S10 = 1 if the file is in an external device
320             **             = 0 if the file is in RAM/ROM
321             **         R0(15,14) = Current position in the file i/o buffer
322             **                     if is an external file
323             **         STMTD1 contains the FIB entry address
324             ** Exit:   D0 @ 1st byte after skipping
325             **
326             ** Calls:  D0+2, POLL(pRDNBF)
327             **
328             ** Uses:  A,C D0, ST[4-0]
329             **
330             ** Stk lvls:  +3
331             ************************************************************************
332             ************************************************************************
333             *
334 139FA CC   =SKPBYT A=A-1  A
335 139FC 400         RTNC
336 139FF 7F20        GOSUB  D0+2RD
337 13A03 66FF        GOTO   SKPBYT
338             *
339             ************************************************************************
340             ************************************************************************
341             **
342             ** Name:(S) RDLNAS  -  Read String Length from a TEXT File.
343             ** Name:    RDLNFX  -  Read String Length from a DATA File.
344             **
345             ** Category:   FILUTL
346             **
347             ** Purpose:  RDLNAS - Read string length from a LIF1 file
348             **           RDLNFX - Read string length from the fixed length
349             **                    file.
350             **
351             ** Entry:   D0 @ current file pointer, absolute addr if file
352             **            in RAM/ROM, absolute address in file I/O buffer
353             **            if file is in external device.
354             **          R0(15,14) = current position in file I/O buffer if
355             **            file is in external device.
356             **          STMTD1 contains FIB entry address
357             ** Exit:    A(A) = The two bytes read from the file
358             **          D0 @ past the two length bytes
359             **          The file pointer in the FIB is not updated. However,
360             **          if the string length is read from the I/O buffer,
361             **          there is a possibility that the I/O bufffer is
362             **          overflowed and the next sector is read into the I/O
363             **          buffer. In this case, if want to back up the D0 by
364             **          two bytes, call the routine BACK2B.
365             **
```

```
366             ** Calls:  RDBYTA
367             **
368             ** Uses:  A, C, D0
369             **
370             ** Stk lvls:  +3
371             ****************************************************************
372             ****************************************************************
373             *
374 13A07 D0    =RDLNFX A=0    A
375 13A09 7220          GOSUB  RDBYTA          READ IN LS BYTE FIRST
376 13A0D 814           ASRC
377 13A10 814           ASRC
378 13A13 7810          GOSUB  RDBYTA          READ IN MS BYTE NEXT
379 13A17 810           ASLC
380 13A1A 810           ASLC
381 13A1D 03            RTNCC
382             *
383 13A1F D0    =RDLNAS A=0    A
384 13A21 7A00          GOSUB  RDBYTA
385 13A25 F0            ASL    A
386 13A27 F0            ASL    A
387 13A29 7200          GOSUB  RDBYTA
388 13A2D 03            RTNCC
389
390             *
391             ****************************************************************
392             ****************************************************************
393             **
394             ** Name:(S) RDBYTA  -  Read Byte From an Opened File Into A
395             ** Name:(S) D0+2RD  -  Move file pointer&check buffer overflow
396             **
397             ** Category:   FILUTL
398             **
399             ** Purpose: Read a byte from an file into A-reg.
400             **          Reading a byte from memory can be easily done by one
401             **          instruction "A=DAT0 B". But if the byte is read from
402             **          an I/O buffer, then the possibility of overflowing
403             **          the I/O buffer should be considered. This routine
404             **          takes care of this problem automatically.
405             **
406             ** Entry:  D0 @ current file pointer(abs.addr. if file in RAM
407             **           or ROM, absolute addr @ file I/O buffer if file in
408             **           external device)
409             **         R0(15,14) = Current byte position in the file I/O
410             **                     buffer if the file is in external device
411             **         STMTD1 contains FIB entry address
412             **         S10 = 0 if file is in RAM or ROM.
413             **             = 1 if file is in an external mass memory device.
414             **
415             ** Exit:  A(B) = The byte
416             **         D0 past the byte.
417             **         R0[15,14] is updated.
418             **         Current position in FIB will be updated if need to
419             **           read in next sector from the external file.
420             **
```

```
421             ** Calls:  DO+2RD, POLL(pRDNBF)
422             **
423             ** Uses:
424             **   Internal file:  Nothing
425             **   External file:  A(14,5), B(15,5), C, D0, R0, ST[4-0]
426             **
427             ** Stk lvls:
428             **   Internal file:  0
429             **   External file:  3
430             **
431             ********************************************************************
432             ********************************************************************
433             *
434             *
435             *
436             ********************************************************************
437             ********************************************************************
438             **
439             ** Name:(S) pRDNBF  -  Write Current Sector, Read Next Sector
440             **
441             ** Category:   POLL
442             **
443             ** Type:       FPOLL
444             **
445             ** Purpose:
446             **      Using the FIB, write current file I/O buffer to
447             **      where it came from in a mass memory device, and read in
448             **      next sector to the file I/O buffer.
449             **
450             **      There are total of 3 polls can be used to read/write a
451             **      sector between a mass memory device and I/O buffer:
452             **      1. pRDNBF - Writes buffer out to current sector and read
453             **                  in next sector. If buffer content has not
454             **                  been altered, just read in next sector.
455             **      2. pRDCBF - Reads in current sector from mass memory
456             **                  device to the I/O buffer. This poll does
457             **                  not care about the content currently in the
458             **                  I/O buffer.
459             **      3. pWRCBF - Writes I/O buffer to current sector in the
460             **                  mass memory device.
461             **
462             **
463             ** Should poll be "Handled" (return with XM=0)?:Yes
464             **
465             ** Meaning of "Handling" Poll (what does code do if handled?):
466             **      As specified above.
467             **
468             ** Entry conditions for handler (registers, ST, RAM, etc.):
469             **      B[A] = Poll number.
470             **      HEX mode.
471             **      P=0.
472             **      STMTD1 contains the FIB entry address of the file
473             **
474             ** Normal exit conditions from handler if handled (ST, RAM,
475             ** registers, etc.):
```

```
476                **      Carry clear.
477                **      HEX mode.
478                **      XM=0.
479                **      Current position in FIB is set to start of next sector.
480                **      File access nib in FIB is set to zero.
481                **
482                **      This poll handler always call the routine SNAPRS to
483                **       restore A,D,D0,D1 from the snap save RAM before return.
484                **      So if the polling routine calls the SNAPSV before issuing
485                **       this poll, it can consider A,D,D0,D1 will not be change
486                **       by this poll handler.
487                **
488                ** Normal exit conditions from handler if not handled (ST, RAM,
489                ** registers, etc.):
490                **      HEX mode.
491                **      XM=1.
492                **
493                ** Error exit conditions from handler:
494                **      Won't return to calling routine if error occur, direct
495                **       exit to BSERR routine.
496                **
497                ** Available subroutine levels: 3
498                **
499                ** What registers/RAM may be used if handled?:
500                **      A-D, D0, D1, P, ST[0-4]
501                **      (B,C,P,ST[0-4] if SNAPSV been called)
502                **
503                ** What registers/RAM may be used if not handled?:
504                **      C, D0
505                **
506                ** History:
507                **
508                **    Date     Programmer              Modification
509                **  --------   ----------   ----------------------------------
510                **  04/20/83   SC           Document
511                **
512                ***************************************************************
513                ***************************************************************
514                *
515 13A2F 14A  =RDBYTA A=DAT0 B
516 13A32 161  =D0+2RD D0=D0+ 2
517 13A35 86A         ?ST=0  I/Obuf
518 13A38 00          RTNYES
519 13A3A 118         C=R0
520 13A3D 812         CSLC
521 13A40 812         CSLC
522 13A43 B66         C=C+1  B              REACHED END OF BUFFER YET ?
523 13A46 564         GONC   RESTR0         IF NOT, PUT BYTE COUNT BACK TO R0
524 13A49 AF9  D0+2RX C=B    W              SAVE B IN R0
525 13A4C 108         R0=C
526 13A4F 07          C=RSTK                Save one level of RSTK
527 13A51 7360        GOSUB  fpoll+
528 13A55 91          CON(2) =pRDNBF
529 13A57 6C70        GOTO   RSREGS
530                *
```

```
531                *
532                ****************************************************************
533                ****************************************************************
534                **
535                ** Name:(S) WRBYTC  -  Write Byte to an Opened File From C
536                ** Name:    WRBYTD  -  Write a Byte to an Opened File
537                **
538                ** Category:   FILUTL
539                **
540                ** Purpose:  Write a byte to a file in RAM/ROM or to a file I/O
541                **           buffer if the file is in external device
542                **
543                ** Entry:  D0 @ current file pointer(absolute addr if file in
544                **           RAM/ROM, absolute @ file I/O buffer if file is in
545                **           external device.
546                **         R0(15,14) = Current byte position in file I/O buffer
547                **           if the file is in an external device.
548                **         S10 = 0 if the file is an internal file
549                **               1 if the file is an external file
550                **
551                **   WRBYTC:  C(B) = The byte to write
552                **   WRBYTD:  D1 @ The byte to write to the byte to be written
553                **
554                ** Exit:  D0 past the source byte.
555                **        For an external file:
556                **          R0(15,14) will be updated.
557                **          If overflow the I/O buffer, current buffer will be
558                **          written back to the file, next sector will be read
559                **          into the I/O buffer, current position in FIB will be
560                **          updated.
561                **
562                **   WRBYTD:  D1 past the byte
563                **
564                ** Calls:  D0+2WR, POLL(pRDNBF)
565                **
566                ** Uses:
567                **   Internal file:  D1
568                **   External file:  A(14,5), B(15,5), C, D0, R0,ST[4-0]
569                **
570                ** Stk lvls:
571                **   Internal file:  0
572                **   External file:  3 if have to flush the I/O buffer.
573                ****************************************************************
574                ****************************************************************
575 13A5B 31FF    WRTEOF LC(2) =fEOF
576 13A5F 6310           GOTO  WRBYTC
577                *
578 13A63 110     WRTMRK A=R0
579 13A66 130           D0=A
580 13A69 6900          GOTO   WRBYTC
581                *
582 13A6D 14F    =WRBYTD C=DAT1 B
583 13A70 171           D1=D1+ 2
584 13A73 14C    =WRBYTC DAT0=C B
585 13A76 161     D0+2WR D0=D0+ 2
```

```
586 13A79 86A          ?ST=0  I/Obuf
587 13A7C 00           RTNYES
588 13A7E 118          C=R0
589 13A81 812          CSLC
590 13A84 812          CSLC
591 13A87 B66          C=C+1  B
592 13A8A 4D0          GOC    D0+2WX        I/O BUFFER FULL
593 13A8D 816  RESTR0  CSRC
594 13A90 816          CSRC
595 13A93 108          R0=C
596 13A96 03           RTNCC
597             *
598 13A98 AF9  D0+2WX  C=B    W              Save B in R0
599 13A9B 108          R0=C
600 13A9E 07           C=RSTK                Save one level of RSTK
601 13AA0 7410         GOSUB  fpoll+
602 13AA4 91           CON(2) =pRDNBF
603 13AA6 136          CDOEX
604 13AA9 06           RSTK=C
605 13AAB 7940         GOSUB  SETWRT         SET ACCESS CODE = 1
606 13AAF 07           C=RSTK
607 13AB1 134          D0=C
608 13AB4 6F10         GOTO   RSREGS
609             *
610 13AB8 8F00  fpoll+ GOSBVL =SNAPSV        Save registers and 1 level of RSTK
          000
611 13ABF 8C00 =fpoll  GOLONG =FPOLL         FAST POLL
          00
612             *
613             *
614             *
615             ********************************************************************
616             ********************************************************************
617             **
618             ** Name:(S) D0=FIB  -  Set D0,C(A) to value at STMTD1
619             **
620             ** Category:   PTRUTL
621             **
622             ** Purpose:
623             **      Set D0,C(A) to the value stored in STMTD1
624             **
625             ** Entry:
626             **
627             ** Exit:
628             **      D0,C(A) = (STMTD1)
629             **
630             ** Calls:      None
631             **
632             ** Uses.......
633             **  Inclusive: D0,C(A)
634             **
635             ** Stk lvls:   0
636             **
637             ** History:
638             **
```

```
                **    Date     Programmer            Modification
                ** --------  ----------  ---------------------------------
                ** 11/06/83   BS          Added documentation
                **
                ********************************************************************
                ********************************************************************
13AC5 1B69 8F2  =DO=FIB DO=(5) =STMTD1
13ACC 146       C=DATO A
13ACF 134       DO=C                    DO = FIB ENTRY ADDRESS
13AD2 03        RTNCC
                *
                ********************************************************************
                ********************************************************************
                **
                ** Name:    RSREGS  -  Restore Registers
                **
                ** Category:   FILUTL
                **
                ** Purpose:  This routine is called after returning from three
                **           polls: pRDCBF, pRDNBF, pWRCBF.
                **           These three poll will not preserve B. So the B
                **           register is saved in R0 before polling and is
                **           restored by this routine.
                **           This routine also sets D0 and R0[A] points to start
                **           of the I/O buffer, sets the byte count in R0[15,14]
                **           to zero.
                ** Entry:  R0= Saved B register.
                **         B(A) = Routine return address
                **
                ** Exit:  Return to the address contained in B(A) on entry
                **        B = R0 on entry
                **        D0 = R0[A] pointe at start of the file I/O buffer
                **        R0(15,14) = 00
                **
                ** Calls: None
                **
                ** Uses:  B,C,D0,R0
                **
                ** Stk lvls:  1
                ********************************************************************
                ********************************************************************
                *
13AD4 D9     RSREGS C=B    A
13AD6 06            RSTK=C
                *
13AD8 136           CD0EX
13ADB D5            B=C    A
13ADD AF2           C=0    W
13AE0 3200 2        LC(3)  512          SET D0 BACK TO START OF BUFR
13AE5 E1            B=B-C  A
13AE7 DD            CBEX   A
13AE9 134           DO=C
13AEC 128           CR0EX
```

```
692 13AEF AF5          B=C    W
693 13AF2 8C00         GOLONG =CHKRSP        SEE IF HPIL MODULE THERE
          00
694                *
695                ****************************************************************
696                ****************************************************************
697                **
698                ** Name:    SETWRT  -  Set Write Access Nib in File's FIB entry
699                **
700                ** Category:   FILUTL
701                **
702                ** Purpose:  Set the access nibble to 1 in the file's FIB entry
703                **           to indicate the content of a file I/O buffer has
704                **           been altered.
705                **    The access nibble is used when:
706                **     1. Need to read in the next sector from the file.
707                **     2. Close the file.
708                **        If this nibble is 1, the I/O buffer need to be written
709                **        back to the file first.
710                **    Every time a new sector is read into the I/O buffer, this
711                **    nibble is zeroed.
712                **
713                ** Entry:  STMTD1 = FIB entry address of the file
714                **         P = 0
715                **
716                ** Exit:  P = 0
717                **
718                ** Calls: D0=FIB
719                **
720                ** Uses:  C(A),D0
721                **
722                ** Stk lvls:  1
723                ****************************************************************
724                ****************************************************************
725                *
726 13AF8 79CF =SETWRT GOSUB  D0=FIB
727 13AFC 16A          D0=D0+ (oACCSb)
728 13AFF 301          LC(1)  1
729 13B02 1540         DAT0=C P
730 13B06 03           RTNCC
731                *
732                ****************************************************************
733                ****************************************************************
734                **
735                ** Name:(S) BACK1B  -  Back up the File Pointer by 1 Byte
736                ** Name:(S) BACK2B  -  Back up the File Pointer by 2 Bytes
737                ** Name:(S) BACK3B  -  Back up the File Pointer by 3 Bytes
738                **
739                ** Category:   FILUTL
740                **
741                ** Purpose:  Sets the current position field of the file's FIB
742                **           back the specified number of bytes. If the new
743                **           position falls in the previous sector, it is read
744                **           into the file's I/O buffer.
745                **
```

```
                ** Entry:  P= 0
                **         R0(15,14) = Current byte pointer in the buffer
                **         R0(4,0) = Current absolute address in the buffer
                **         S10 = 0 - Internal file
                **               1 - External file
                **         STMTD1 contains file FIB address
                **
                **
                ** Exit:  P = 0
                **
                ** Calls:  POLL(pRDCBF)
                **
                ** Uses:  A,B,C,D0,P
                **
                ** Stk lvls:  0 - internal file
                **           4 - external file (if has to back up)
                ****************************************************************
                ****************************************************************
                *
13B08 0C        =BACK3B P=P+1
13B0A 0C        =BACK2B P=P+1
13B0C 0C        =BACK1B P=P+1
                *
13B0E 87A   CKST10 ?ST=1  I/Obuf
13B11 60           GOYES  CKBPOS
13B13 20           P=     0
13B15 02           RTNSC
                *
13B17 118   CKBPOS C=R0
13B1A 812          CSLC
13B1D 812          CSLC
13B20 A6E   CKBF10 C=C-1  B
13B23 4C0          GOC    BACKUP
13B26 0D           P=P-1
13B28 890          ?P=    0
13B2B 00           RTNYES
13B2D 52F          GONC   CKBF10
                *
13B30 20    BACKUP P=     0
                *
                * HAS TO MOVE THE CURRENT POSITION BACKWARD 1 SECTOR
                *
13B32 7F8F         GOSUB  D0=FIB
13B36 D2           C=0    A
13B38 3182         LC(2)  =oCPOSb
13B3C 132          ADOEX
13B3F C2           C=C+A  A
13B41 134          D0=C
13B44 146          C=DAT0 A
13B47 81E          CSRB
13B4A 816          CSRC
13B4D 816          CSRC
13B50 CE           C=C-1  A
13B52 812          CSLC
13B55 812          CSLC
```

```
 801 13B58 C6          C=C+C  A
 802 13B5A 144         DAT0=C A
 803 13B5D 07          C=RSTK                Save one level of RSTK in SNAPE
 804 13B5F 755F        GOSUB  fpoll+
 805 13B63 81          CON(2) =pRDCBF
 806 13B65 D9          C=B    A
 807 13B67 06          RSTK=C
 808 13B69 03          RTNCC
 809             *
 810             *
 811             ************************************************************
 812             ************************************************************
 813             **
 814             ** Name:    REDREC  -  Read one Record from the TEXT File
 815             **
 816             ** Category:   FILUTL
 817             **
 818             ** Purpose:  Read in the next record in the TEXT file and put
 819             **           it on top of stack
 820             **
 821             ** Entry:  D1 @ Top of stack
 822             **         R0(4,0) = File pointer (@ string length)
 823             **         R(15,14) = Byte pointer in buffer if external file
 824             **         S10 = 1 - If read from I/O buffer
 825             **               0 - If read from RAM?ROM
 826             **
 827             ** Exit:   If carry set => reached end of file
 828             **         D1 @ Top of stack
 829             **         R0 = RAM address past the string
 830             **         Current positoin in the FIB has been updated
 831             ** Uses:  A,B,C,D0,D1
 832             **
 833             ** Stk lvls: 1 if file is in RAM or ROM.
 834             **           5 if file is in external device.
 835             ************************************************************
 836             ************************************************************
 837             *
 838 13B6B 110   =REDREC A=R0
 839 13B6E 130         D0=A
 840 13B71 7AAE        GOSUB  RDLNAS         READ STRING LENGTH
 841             *
 842 13B75 23          P=     3
 843 13B77 B14         A=A+1  WP
 844 13B7A 5A0         GONC   RREC20
 845 13B7D 20          P=     0
 846 13B7F 778F        GOSUB  BACK2B
 847 13B83 02          RTNSC
 848             *
 849             *
 850 13B85 A1C   RREC20 A=A-1  WP
 851 13B88 D1          B=0    A
 852 13B8A A98         B=A    WP             B(A) = RECORD LENGTH
 853 13B8D 20          P=     0
 854 13B8F 301         LC(1)  1
 855 13B92 0E05        C=B&C  P
```

```
856 13B96 90A          ?C=0   P              LENGTH EVEN ?
857 13B99 50           GOYES  RREC25
858 13B9B 856          ST=1   Odd
859              *
860 13B9E D9    RREC25 C=B    A
861 13BA0 C6           C=C+C  A              C= RECORD LENGTH IN NIBS
862 13BA2 1CF          D1=D1- 16
863 13BA5 133          AD1EX
864 13BA8 EA           A=A-C  A
865 13BAA 7B02         GOSUB  STKSPC
866 13BAE 4C0          GOC    RREC30
867 13BB1 07           C=RSTK                POP OFF RETURN ADDRESS
868 13BB3 735F         GOSUB  BACK2B         NEED TO READ BACK PREVIOUS RECORD?
869 13BB7 60C2         GOTO   prt185         NOT ENOUGH ROOM ON STACK
870              * GENERATES THE STRING HEADER ON STACK
871 13BBB 31F0  RREC30 LCHEX  0F
872 13BBF 14D          DAT1=C B
873 13BC2 171          D1=D1+ 2
874 13BC5 AF2          C=0    W
875 13BC8 D9           C=B    A
876 13BCA C6           C=C+C  A
877 13BCC 15DD         DAT1=C 14             WRITE STRING LENGTH
878 13BD0 17D          D1=D1+ 14
879 13BD3 133          AD1EX
880 13BD6 CA           A=A+C  A
881 13BD8 131          D1=A                  D1 @ PAST THE STRING
882 13BDB D4           A=B    A
883 13BDD CC    RREC40 A=A-1  A
884 13BDF 431          GOC    RREC50
885 13BE2 14E          C=DAT0 B
886 13BE5 1C1          D1=D1- 2
887 13BE8 14D          DAT1=C B
888 13BEB 734E         GOSUB  D0+2RD
889 13BEF 6DEF         GOTO   RREC40
890 13BF3 1CF   RREC50 D1=D1- 16             D1 @ TOP OF STACK
891 13BF6 866          ?ST=0  Odd
892 13BF9 60           GOYES  RREC60
893 13BFB 733E         GOSUB  D0+2RD
894 13BFF 6760  RREC60 GOTO   UPCPOS
895              *
896              ********************************************************************
897              ********************************************************************
898              **
899              ** Name:    REDNUM  -  Read 8-Byte Number From File
900              **
901              ** Category:   FILUTL
902              **
903              ** Purpose:  Read a number (8 bytes) from DATA or SDATA file.
904              **
905              ** Entry:  R0 = Current file pointer
906              **         D(A) = # of bytes to end of file
907              **         D1 @ top of the math stack where the number will be
908              **          stored (D1 to D1+15 should be free).
909              **         S10 = 0 if file is in RAM or ROM
910              **             = 1 if file is in external mass memory device.
```

```
911             **            R0(15,14) = byte pointer in I/O buffer
912             **
913             ** Exit:  D1 unchanged and is pointing at the number.
914             **        C(A)= 8
915             **        If reached end of file, will exit to EOFIL
916             **        If the sign of the number is not 0 or 9, will exit
917             **           to MFERR ("Data Type").
918             **
919             ** Used:  A,C,D0
920             **
921             ************************************************************
922             ************************************************************
923             *
924 13C03 110  =REDNUM A=R0
925 13C06 130         D0=A
926             ************************************
927             * Pack here could save 1 nibble
928             *
929             * RDNUM0 GOSUB  LDC=8
930             *
931 13C09 D2    RDNUM0 C=0    A
932 13C0B 308         LC(1)  8
933             *
934             * End of possible packing
935             ************************************
936             *
937 13C0E E3          D=D-C  A
938 13C10 560         GONC   RDNUM1
939 13C13 66D3        GOTO   EOFIL
940             * READ THE 8 BYTES FROM FILE AND WRITE IT TO STACK FIRST
941             *
942 13C17 A8A   RDNUM1 A=C    P              A(0) = 8
943 13C1A A0C   RDNUM2 A=A-1  P
944 13C1D 431         GOC    RDNUM5
945 13C20 14E         C=DAT0 B              READ A BYTE FROM FILE,
946 13C23 14D         DAT1=C B              WRITE A BYTE TO STACK
947 13C26 171         D1=D1+ 2
948 13C29 750E        GOSUB  D0+2RD
949 13C2D 6CEF        GOTO   RDNUM2
950             *
951             * READ THE NUMBER FROM STACK AND REFORM IT.
952             *
953 13C31 1CF   RDNUM5 D1=D1- 16
954 13C34 14B         A=DAT1 B
955 13C37 171         D1=D1+ 2
956 13C3A 1534        A=DAT1 S
957 13C3E 170         D1=D1+ 1
958 13C41 1532        A=DAT1 XS
959 13C45 170         D1=D1+ 1
960 13C48 1535        A=DAT1 M
961 13C4C 1C3         D1=D1- 4
962 13C4F 1517        DAT1=A W              WRITE THE NUMBER BACK TO STACK
963 13C53 948         ?A=0   S              SIGN =0 ?
964 13C56 11          GOYES  UPCPOS
965 13C58 2F          P=     15
```

```
 966 13C5A 309          LC(1)  9              SIGN = 9 ?
 967 13C5D 20           P=     0
 968 13C5F 942          ?A=C   S
 969 13C62 50           GOYES  UPCPOS
 970 13C64 855          ST=1   Error
 971             *
 972             *
 973             ****************************************************************
 974             ****************************************************************
 975             **
 976             ** Name:(S) UPCPOS  -  Update FIB Current Position
 977             **
 978             ** Category:   FILUTL
 979             **
 980             ** Purpose:  Update current position in FIB
 981             **
 982             ** Entry:  D0 = Current file pointer or buffer pointer
 983             **         R0(15,14)= Byte pointer in buffer if external file
 984             **         R1(A) = Record length if fixed length data file
 985             **         S9 = 1 for IRAM
 986             **         S10 = 0/1 for internal/external file
 987             **         S11 = 0/1 for serial/random access
 988             **         STMTD1 = Entry address in FIB
 989             **
 990             ** Exit:  Update current position in FIB
 991             **        The D0 on entry is saved in R0(4,0)
 992             **        If is DATA file (copy code = 1) :
 993             **         Carry set => The file pointer is at the beginning
 994             **           of a record and the random access flag is set (S9).
 995             **         A(A) = Number of bytes left in current record.
 996             **         B(A) = Byte position in current record.
 997             **
 998             ** Calls:  IDIV
 999             **
1000             ** Used:  A,B,C,D0,R0,P (B is used only for DATA file)
1001             **
1002             ** Stk lvls:  1
1003             **
1004             ****************************************************************
1005             ****************************************************************
1006             *
1007 13C67 118  =UPCPOS C=R0
1008 13C6A 136         CD0EX
1009 13C6D 108         R0=C
1010 13C70 715E        GOSUB  D0=FIB
1011 13C74 169         D0=D0+ (oCOPYb)
1012 13C77 1564        C=DAT0 S              READ COPY CODE TO C(S)
1013 13C7B 16A         D0=D0+ (oDBEGb)-(oCOPYb)
1014 13C7E 146         C=DAT0 A              C(A) = DATA BEGIN
1015 13C81 16E         D0=D0+ (oRECLb)-(oDBEGb)
1016 13C84 163         D0=D0+ (oCPOSb)-(oRECLb)
1017 13C87 110         A=R0
1018 13C8A 87A         ?ST=1  I/Obuf
1019 13C8D B0          GOYES  UCP010
1020 13C8F EA          A=A-C  A              OFFSET FROM DATA BEGIN IN NIBBLES
```

```
1021 13C91 140          DATO=A A
1022 13C94 6810         GOTO   UCP020          NOT FIXED LENGTH DATA FILE
1023             * EXTERNAL FILE
1024 13C98 146   UCP010 C=DATO A               C= CURRENT POSITION
1025 13C9B 81E          CSRB                   CURRENT POSITION IB BYTES
1026 13C9E 810          ASLC
1027 13CA1 810          ASLC                   A(0,1)= BYTE POSITION IN BUFFER
1028 13CA4 AE6          C=A    B
1029 13CA7 A76          C=C+C  W               C= CURRENT POSITION IN NIBBLES
1030 13CAA 144          DATO=C A
1031             *
1032 13CAD A4E   UCP020 C=C-1  S               SEE IF IS FIXED LENGTH DATA FILE
1033 13CB0 A4E          C=C-1  S               C= RECORD LENGTH
1034 13CB3 4A0          GOC    UCP025
1035 13CB6 118          C=R0
1036 13CB9 134          D0=C
1037 13CBC 03           RTNCC
1038             * FIX LENGTH DATA FILE
1039 13CBE AF2   UCP025 C=0    W
1040 13CC1 111          A=R1
1041 13CC4 D6           C=A    A
1042 13CC6 AF0          A=0    W
1043 13CC9 142          A=DATO A
1044 13CCC 81C          ASRB                   CURRENT POSITION IN BYTES
1045 13CCF 8E00         GOSUBL =IDIV
          00
1046 13CD5 20           P=     0
1047 13CD7 111          A=R1   A               A(A) = RECORD LENGTH
1048 13CDA 8A9          ?B=0   A               **Pack here...could save 5 nibs***
1049 13CDD 40           GOYES  UCP030
1050 13CDF E0           A=A-B  A               # OF BYTE TO END OF RECORD
1051 13CE1 16B   UCP030 D0=D0+ (oRLENb)-(oCPOSb)
1052 13CE4 140          DATO=A A
1053 13CE7 118          C=R0                   **Pack here...could save 2 nibs***
1054 13CEA 134          D0=C                   RESTORE D0
1055 13CED 119          C=R1
1056 13CF0 8A6          ?C#A   A               AT BEGINNING OF NEXT RECORD ?
1057 13CF3 70           GOYES  UCP040          IF NOT, RETURN WITH CARRY CLEAR
1058 13CF5 879          ?ST=1  Random          RANDOM ACCESS ?
1059 13CF8 00           RTNYES                 IF SO, RETURN WITH CARRY SET
1060 13CFA 03    UCP040 RTNCC
1061             *
1062             ************************
1063             * EORCHK - IN RANDOM ACCESS, IF A DATA ITEM IS ENDED AT THE LAST
1064             *          BYTE OF A RECORD, IT IS HARD TO DETECT THE NEXT DATA
1065             *          ITEM, IF ANY, THAT IS CROSSING THE RECORD BOUNDARY.
1066             *          THIS ROUTINE IS JUST FOR THIS PURPOSE.
1067             * ENTRY: (STMTD1) CONTAINS FIB ENTRY ADDRESS OF THE FILE
1068             *        R1(A) = RECORD SIZE
1069             * EXIT:  CARRY SET IF THE NEXT DATA ITEM IS CROSSING RECORD
1070             *        BOUNDARY.
1071             * USES:  A(A), C(C), D0, D1
1072             *
1073 13CFC 869   EORCHK ?ST=0  Random
1074 13CFF BF           GOYES  UCP040            RETURN AND CLEAR CARRY
```

```
1075 13D01 1B69          D0=(5) =STMTD1
           8F2
1076 13D08 142           A=DAT0 A
1077 13D0B 3443          LC(5)  (oRLENb)        ***Pack here...could save 1 nib**
           000
1078 13D12 CA            A=A+C  A
1079 13D14 130           D0=A
1080 13D17 142           A=DAT0 A
1081 13D1A 119           C=R1
1082 13D1D 8A2           ?A=C   A
1083 13D20 40            GOYES  EORC20          **Pack here...could save 2 nibs***
1084 13D22 03            RTNCC
1085               * NOW THE FILE POINTER IS AT BEGINNING OF A RECORD
1086               * SEE IF THE PRINT LIST JUST EXHAUSTED
1087 13D24 841    EORC20 ST=0   Array
1088 13D27 848           ST=0   Fresh
1089 13D2A 7401          GOSUB  CHKDON
1090 13D2E 483           GOC    EOLCHK
1091               *
1092 13D31 8E5D          GOSUBL STKVCT
           90
1093 13D37 861           ?ST=0  Array
1094 13D3A 90            GOYES  EORC30
1095 13D3C 8AA           ?C=0   A
1096 13D3F 82            GOYES  EOLCHK
1097 13D41 02            RTNSC
1098 13D43 8E00   EORC30 GOSUBL =POPMTH
           00
1099 13D49 133           AD1EX
1100 13D4C 1FE9          D1=(5) =FORSTK
           5F2
1101 13D53 147           C=DAT1 A
1102 13D56 8A2           ?A=C   A
1103 13D59 E0            GOYES  EOLCHK
1104 13D5B 131           D1=A
1105 13D5E 8E3E          GOSUBL STKV15
           90
1106 13D64 400           RTNC
1107               *
1108               ********************************************************************
1109               ********************************************************************
1110               ** Name:(S) CHKEOL  -  Check if at End of Statement
1111               **
1112               ** Category: EXCUTL
1113               **
1114               ** Purpose: When processing the PRINT or READ list, check
1115               **          to see if just past the last variable on the
1116               **          list.
1117               **
1118               ** Entry: D0 = Program counter
1119               **
1120               ** Exit: Carry set => Not at end of statement yet.
1121               **       Carry clear => PC is at end of the statement
1122               **
1123               ** Uses: A(B), C(B)
```

```
1124            **
1125            ** Stk lvls : 0
1126            ****************************************************************
1127            ****************************************************************
1128            **
1129 13D67 8E00  EOLCHK GOSUBL =RESTDO
           00
1130            *
1131 13D6D 14A   =CHKEOL A=DAT0 B
1132 13D70 3100         LC(2)  =tCOMMA
1133 13D74 962          ?A=C   B
1134 13D77 00           RTNYES
1135 13D79 B66          C=C+1  B              C(B) = tSEMIC
1136 13D7C 962          ?A=C   B
1137 13D7F 00           RTNYES
1138 13D81 03           RTNCC
1139            *
1140            ****************************************************************
1141            * IF MAINFRAME FILE AND COPY CODE<8, RETURN WITH CARRY CLEAR
1142 13D83 7A64  CHKCOD GOSUB  saved0
1143 13D87 7610         GOSUB  MFLG=0
1144 13D8B 8E5A         GOSUBL GTPTRS
           80
1145 13D91 ACB          C=D    S
1146 13D94 A46          C=C+C  S              IF COPY CODE >= 8, POLL
1147 13D97 400          RTNC
1148 13D9A 94A          ?C=0   S
1149 13D9D 00           RTNYES
1150 13D9F 03           RTNCC
1151            *
1152            ****************************************************************
1153            ****************************************************************
1154            **
1155            ** Name:(S) MFLG=0  -  Clear MLFFLG nibble
1156            ** Name:    MFLG=X  -  Set MLFFLG nibble
1157            **
1158            ** Category:   GENUTL
1159            **
1160            ** Purpose:
1161            **      MFLG=0:  Clear MLFFLG nibble
1162            **      MFLG=X:  Set MLFFLG nibble
1163            **
1164            ** Entry:
1165            **      MFLG=X:  C(P) is value to be stored at MLFFLG
1166            **
1167            ** Exit:
1168            **      MFLG=0:  C(A)=0
1169            **      (MLFFLG) = Specified value
1170            **
1171            ** Calls:      None
1172            **
1173            ** Uses.....
1174            **  Inclusive: D1,(MLFFLG),  and MFLG=0 entry also uses C(A)
1175            **
1176            ** Stk lvls:   0
```

```
1177            **
1178            ** History:
1179            **
1180            **    Date     Programmer             Modification
1181            **  --------   ----------   ----------------------------------
1182            **  11/06/83   BS           Added documentation
1183            **
1184            ********************************************************************
1185            ********************************************************************
1186 13DA1 D2    =MFLG=0 C=0    A              Clear user fcn sticky nibble.
1187 13DA3 1F07 =MFLG=X D1=(5) =MLFFLG
         8F2
1188 13DAA 1550          DAT1=C P
1189 13DAE 01            RTN
1190            *
1191            ********************************************************************
1192            ********************************************************************
1193            **
1194            ** Name:  CHKSTK  -  Check Free Space on Math Stack
1195            ** Name:  STK16?  -  Check Free Space on Math Stack for 16 Nibs
1196            ** Name:  STK19?  -  Check Free Space on Math Stack for 19 Nibs
1197            **
1198            ** Category:   SAVSTK
1199            **
1200            ** Purpose:  Check if still room left in the math stack
1201            **
1202            ** Entry:  D1 pts current top of stack
1203            **
1204            ** Exit:  A = D1
1205            **      Carry set => O.K.
1206            **      Carry clear => D1 ran into AVMEMS
1207            **
1208            ** Calls:  none
1209            **
1210            ** Uses:  A(A), C(A)
1211            **        D1 by STK16? and STK19?
1212            **
1213            ** Stk lvls:  0
1214            ********************************************************************
1215            ********************************************************************
1216            *
1217 13DB0 1C2  =STK19? D1=D1- 3
1218 13DB3 1CF  =STK16? D1=D1- 16
1219 13DB6 133  =CHKSTK AD1EX
1220 13DB9 1F49  STKSPC D1=(5) =AVMEMS
         5F2
1221 13DC0 147           C=DAT1 A
1222 13DC3 E2            C=C-A  A
1223 13DC5 131           D1=A                    D1= NEW STACK POINTER
1224 13DC8 01            RTN                     CARRY SET - OK
1225            *
1226            ********************************************************************
1227            *
1228            * ENTRY: D1 @ STRING LENGTH(2 NIBS PAST STRING HEADER)
1229            *
```

```
1230            * EXIT:  C=STRING LENGTH IN NIBBLES
1231            *        D1 @ END OF STRING ON STACK
1232            *
1233 13DCA 171   DROP02 D1=D1+ 2
1234            *
1235 13DCD AF2   DROPST C=0    W
1236 13DD0 147          C=DAT1 A              C=STRING LENGTH
1237 13DD3 17D          D1=D1+ 14
1238 13DD6 133          AD1EX
1239 13DD9 CA           A=A+C  A              A @ PAST STRING ON STACK
1240            *
1241 13DDB 1F99  D1UPMT D1=(5) =MTHSTK
          5F2
1242 13DE2 141          DAT1=A A
1243 13DE5 131          D1=A
1244 13DE8 03           RTNCC
1245            *
1246            *
1247            *
1248 13DEA 171   UPMH34 D1=D1+ 2
1249 13DED 17F          D1=D1+ 16
1250 13DF0 17F   UPMH16 D1=D1+ 16
1251 13DF3 04           SETHEX
1252 13DF5 133   UPMTHS AD1EX
1253 13DF8 62EF         GOTO   D1UPMT
1254            *
1255            ************************************************************
1256            ************************************************************
1257            **
1258            ** Name:    D1=MST+  -  Set D1 to Math Stack Pointer
1259            **          and clear all status bits, set S8
1260            **
1261            ************************************************************
1262            ************************************************************
1263            *
1264 13DFC 1B07  CKMFLG D0=(5) =MLFFLG
          8F2
1265 13E03 14E          C=DAT0 B
1266 13E06 90A          ?C=0   P
1267 13E09 81           GOYES  D1MST+
1268 13E0B D2           C=0    A
1269 13E0D 1540         DAT0=C P
1270 13E11 1FF6         D1=(5) =CHN#SV
          9F2
1271 13E18 14B          A=DAT1 B
1272 13E1B 8E00         GOSUBL =FIBADR
          00
1273 13E21 77D7 =D1MST+ GOSUB  D1mstk
1274 13E25 08           CLRST
1275 13E27 858          ST=1   Fresh
1276 13E2A 03           RTNCC
1277            *
1278            *
1279 13E2C      =POlljj
1280 13E2C 8C00  poll   GOLONG =POLL
```

```
          00
1281            *
1282            ****************************************************************
1283            * Name:  CHKDON - Check if done with one variable
1284            * Purpose : When executing the READ# or PRINT# statement, if
1285            *   the next item on the list is an array variable, its dope
1286            *   vector will be saved on the math stack and the value of
1287            *   next element will be recalled to on top of the dope vector.
1288            *   Then when we done with one data item, we pop the math stack
1289            *   and check what is left on the math stack. If the math stack
1290            *   pointer is matching the FOR stack point, we know we done with
1291            *   one variable or expression on the list. If the dope vector
1292            *   is still left on the stack, we know we are working with an
1293            *   array. So we will decrement the element counter and point to
1294            *   next element in the array if the count is zero yet.
1295            *
1296            * Entry: AVMEME and FORSTK
1297            *
1298            * EXIT:  CARRY SET => DONE WITH ONE ITEM
1299            *        CARRY CLEAR => NOT DONE YET
1300            *
1301 13E32 1FE9 =CHKDON D1=(5) =FORSTK
          5F2
1302 13E39 147          C=DAT1 A
1303 13E3C 1C4          D1=D1- 5
1304 13E3F 143          A=DAT1 A
1305 13E42 131          D1=A
1306 13E45 8A2          ?A=C   A
1307 13E48 00           RTNYES
1308 13E4A 01           RTN
1309            *
1310            *
1311            ****************************************************************
1312            ****************************************************************
1313            **
1314            ** Name:    NXTVAR  -  Get next Variable from READ list
1315            ** Name:(S) NXTVA-  -  Get next Variable from READ list
1316            **
1317            ** Category:   EXCUTL
1318            **
1319            ** Purpose: Get the next variable from the READ list, the
1320            **          variable will be created if it does not yet exist.
1321            **
1322            ** Entry: D0 @ the next variable token
1323            **
1324            ** Exit: The updated D0 (past the variable) saved in STMTD0
1325            **       MTHSTK is set to current top of stack.
1326            **       The variable value or its dope vector is on top of
1327            **        math stack.
1328            **       DEST has been called ( DEST will save all the
1329            **         information in STMTR0 & STMTR1 that need to assign a
1330            **         value from math stack to the variable).
1331            **
1332            ** Calls: EXPEX-
1333            **
```

```
1334                ** Uses:  All CPU registers, scratch RAM and status.
1335                **
1336                ** Stk lvls: 5
1337                **
1338                ********************************************************************
1339                ********************************************************************
1340                *
1341 13E4C 8E00 =NXTVAR GOSUBL =SVTRC          SAVE PC FOR TRACE
             00
1342 13E52 8E00         GOSUBL =EXPEX-
             00
1343 13E58 7593 =NXTVA- GOSUB  saved0
1344 13E5C 8E00         GOSUBL =DEST
             00
1345 13E62 D2           C=0    A
1346 13E64 B56          C=C+1  M
1347 13E67 8B9          ?B>=C  A              VARIABLE EXIST ?
1348 13E6A 00           RTNYES
1349 13E6C AF9          C=B    W
1350 13E6F 8E00         GOSUBL =NEWVAR        CREATE THE VARIABLE FIRST
             00
1351 13E75 560          GONC   NXTV10
1352 13E78 6355  prt185 GOTO   PRT185         SAY "NO ROOM"
1353 13E7C 1B08  NXTV10 D0=(5) =S-R0-3
             8F2
1354 13E83 1544         DAT0=C S              WRITE VARIABLE TYPE
1355 13E87 908          ?A=0   P              SIMPLE NUMERIC VARIABLE ?
1356 13E8A 00           RTNYES                IF SO, RETURN
1357 13E8C B04          A=A+1  P
1358 13E8F A64          A=A+A  B
1359 13E92 96C          ?A#0   B              A(B) = F8 OR F0 ?
1360 13E95 11           GOYES  NXTV20         IF NOT, GOTO NXTV20
1361 13E97 AF2          C=0    W              PUT F000000000000200 ON STACK
1362 13E9A A0E          C=C-1  P
1363 13E9D 22           P=     2
1364 13E9F 80FD         CPEX   13
1365 13EA3 411          GOC    NXTV30         (B.E.T.)
1366 13EA6 D9    NXTV20 C=B    A              WRITE ARRAY ADDRESS TO STACK
1367 13EA8 134          D0=C
1368 13EAB 1567         C=DAT0 W
1369 13EAF 8E00         GOSUBL =RECADR
             00
1370 13EB5 1557  NXTV30 DAT1=C W
1371 13EB9 02           RTNSC
1372                *
1373                ********************************************************************
1374                ********************************************************************
1375                **
1376                ** Name:    READ#   -  READ# statement execute
1377                **
1378                ** Category: STEXEC
1379                **
1380                ** Purpose:  Read in values from file and store to variables.
1381                **           This code works for file both in RAM and in mass
1382                **           memory device in HPIL.
```

```
1383            **          Only works for file type TEXT, SDATA, DATA, will
1384            **          poll for unrecognized.
1385            **
1386            ** Entry: DO points past the READ# token
1387            **
1388            ** Exit: Exit to NXTSTM
1389            **
1390            ** Detail:
1391            **
1392            **  1. Find the channel # and its FIB entry address, save the
1393            **     FIB entry address in STMTD1.
1394            **  2. If record # is specified, set file pointer points to that
1395            **     record.
1396            **  3. Call expression execution to recall the next variable in
1397            **     the list. If the variable is not exist yet, create it.
1398            **  4. Call the routine DEST to save the destination address in
1399            **     the STMTR0 & STMTR1.
1400            **  5. If the variable is a simple variable, set the MTHSTK to
1401            **     equal to the FORSTK.
1402            **  6. If the variable is an array variable, modify the dope
1403            **     vector and put it on top of the FORSTK and set the MTHSTK
1404            **     to point to it. The modified dope vector contains the #
1405            **     of the elements in the array and the address of the next
1406            **     e;ement.
1407            **  7. Read next data item from the file and update the file
1408            **     pointer.
1409            **  8. Put the value on to the math stack and set the MTHSTK to
1410            **     point to it. Call STORE to assign the value to the
1411            **     variable or array element.
1412            **  9. Drop the stack pointer and set the MTHSTK to pass the
1413            **     data item on the stack.
1414            ** 10. If the updated MTHSTK = FORSTK, we know we just done with
1415            **     a simple variable. Then go to see if the variable
1416            **     exhausted Yet, else continue.
1417            ** 11. If the updated MTHSTK # FORSTK, last data item was
1418            **     assigned to an array element. Decrement the element count
1419            **     in the array dope vector and update the next element
1420            **     address. If the element count drops to zero, done with
1421            **     the array, else continue to next element.
1422            **
1423            ***************************************************************
1424            ***************************************************************
1425            *
1426            ***************************************************************
1427            ***************************************************************
1428            **
1429            ** Name:(S) pREAD#  -  READ# on File of Copycode = 8
1430            **
1431            ** Category:   POLL
1432            **
1433            ** Type:       POLL
1434            **
1435            **
1436            ** Purpose:
1437            **      Execution of READ # statement when the copy code of the
```

```
1438            **      file is 8.
1439            **
1440            ** Should poll be "Handled" (return with XM=0)?:Yes
1441            **
1442            ** Meaning of "Handling" Poll (what does code do if handled?):
1443            **      Complete the execution of READ # statement.
1444            **
1445            ** Entry conditions for handler (registers, ST, RAM, etc.):
1446            **      B[A] = Poll number.
1447            **      HEX mode.
1448            **
1449            **      D[S] = Copy code of the file.
1450            **      D(A) = # of bytes to end of file
1451            **      R0(A)= Current position (absolute address)
1452            **      R0(15:14) = Relative position in buffer if external
1453            **      R1   = Record length in bytes
1454            **      CHN#SV = Channel # specified in the statemnt.
1455            **      STMTD1 = FIB entry address of the file.
1456            **       (All the file related information can be found in the
1457            **        FIB entry of the file)
1458            **      STMTD0 = Program counter points at the semmicolon of the
1459            **               statement.
1460            **      S9  = 0 if serial access (record # not specified)
1461            **          = 1 If random access
1462            **      S10 = 0 if file is in mainframe RAM/ROM
1463            **          = 1 if file is in external mass memory deivce
1464            **      S11 = 0 if file is not in Independent RAM
1465            **          = 1 if file is in Independent RAM
1466            **
1467            **      At the time when this poll is issued, the READ#
1468            **      already process the channel number and the record number
1469            **      -if specified.
1470            **      If the record number has been specified, the pSREC# poll
1471            **      should been issued earlier so the file pointer (in the
1472            **      FIB) should already pointing at the start of the record.
1473            **
1474            ** Normal exit conditions from handler if handled (ST, RAM,
1475            ** registers, etc.):
1476            **      If the poll is handled, the handler should handle the
1477            **      statement completely. So the handler should directly
1478            **      exit to NXTSTM. The handler doesn't need to worry
1479            **      about the math stack used by the POLL routine, it will
1480            **      be taken cared by the run loop.
1481            **
1482            ** Normal exit conditions from handler if not handled (ST, RAM,
1483            ** registers, etc.):
1484            **      Carry clear.
1485            **      HEX mode.
1486            **      XM=1.
1487            **
1488            ** Error exit conditions from handler:
1489            **      Carry set.
1490            **      HEX mode.
1491            **      C[0-3] = Error number.
1492            **
```

```
1493            ** Available subroutine levels: 6
1494            **
1495            ** What registers/RAM may be used if handled?:
1496            **      All CPU registers, scratch RAM, S11-0
1497            **
1498            ** What registers/RAM may be used if not handled?:
1499            **      A, C D0, D1
1500            **
1501            ** What registers/RAM may be used if error exit ?:
1502            **      All CPU registers, scratch RAM, S11-0
1503            **
1504            ** History:
1505            **
1506            **    Date     Programmer          Modification
1507            **  --------   ----------   ----------------------------------
1508            **  04/20/83   SC           Document
1509            **
1510            ************************************************************************
1511            ************************************************************************
1512            *
1513            ************************************************************************
1514            ************************************************************************
1515            **
1516            ** Name:(S) pEOFIL  -  Poll at End-of-File
1517            **
1518            ** Category:   POLL
1519            **
1520            ** Type:       FPOLL
1521            **
1522            ** Purpose:
1523            **      When end of file has been reached in a READ # statement,
1524            **      poll to give a LEX file a chance to act before the READ#
1525            **      statement would otherwise exit to error.
1526            **      One possible thing an LEX file can do is to implement
1527            **      the "ON EOF GOTO/GOSUB <label>" mechanism.
1528            **
1529            ** Should poll be "Handled" (return with XM=0)?:Yes
1530            **
1531            ** Meaning of "Handling" Poll (what does code do if handled?):
1532            **      The end-of-file error has been intercepted.
1533            **
1534            ** Entry conditions for handler (registers, ST, RAM, etc.):
1535            **      Carry set.
1536            **      B[A] = Poll number.
1537            **      HEX mode.
1538            **      P=0.
1539            **      STMTD1 contains the FIB entry address of the file.
1540            **      The file pointer in FIB is pointing at :
1541            **      TEXT file : End-of-file mark (FFFF).
1542            **      SDATA file: Past the last data item of the file.
1543            **      DATA file : Pointing at an end-of-file mark or past the
1544            **                  end of the file.
1545            **
1546            ** Normal exit conditions from handler if handled (ST, RAM,
1547            ** registers, etc.):
```

```
1548                ** If handle, the handler should never return to the polling
1549                ** routine. If it ever returns to the polling routine, an
1550                ** "End of File" will be generated.
1551                ** This poll is just provide a hook for an LEX file
1552                ** to intercept the end-of-file error. The possible thing
1553                ** an LEX file can do to answer this poll is to implement
1554                ** a "ON EOF GOTO/GOSUB <label>" type of trap.
1555                **
1556                ** Normal exit conditions from handler if not handled:
1557                **      HEX mode.
1558                **      P = 0
1559                **
1560                ** Error exit conditions from handler (POLL only):
1561                **      HEX mode.
1562                **      P = 0
1563                **
1564                ** Available subroutine levels: 6
1565                **
1566                ** What registers/RAM may be used if handled?:
1567                **      All CPU registers, scratch RAM, ST11-0.
1568                **
1569                ** What registers/RAM may be used if not handled?:
1570                **      All CPU registers, scratch RAM, ST11-0.
1571                **
1572                ** History:
1573                **
1574                **    Date     Programmer              Modification
1575                ** --------   ----------   ------------------------------------
1576                ** 04/20/83   SC           Document
1577                **
1578                ************************************************************
1579                ************************************************************
1580                *
1581                *
1582 13EBB 161   =READ# D0=D0+  2
1583 13EBE 8E00         GOSUBL  =GETCH#          GET CHANNEL #
           00
1584 13EC4 8E00         GOSUBL  =GETRE#          GET RECORD #
           00
1585                * CHANNEL # IS SAVED IN CHN#SV
1586                *
1587 13ECA 75BE          GOSUB  CHKCOD
1588 13ECE 5C0           GONC   RED150
1589 13ED1 775F          GOSUB  poll
1590 13ED5 72            CON(2) =pREAD#
1591 13ED7 65D3          GOTO   PRT100
1592 13EDB 788E  RED150  GOSUB  EOLCHK
1593 13EDF 460           GOC    RED200
1594 13EE2 61A4          GOTO   PRT140
1595                *
1596 13EE6 161    RED200 D0=D0+ 2
1597 13EE9 7F5F          GOSUB  NXTVAR
1598 13EED 7B0F          GOSUB  CKMFLG          SEE IF USER FCN BEEN CALLED
1599 13EF1 6C00          GOTO   RED236
1600                *
```

```
1601 13EF5 793F  RED230 GOSUB  CHKDON
1602 13EF9 41E   RED235 GOC    RED150
1603 13EFC 08           CLRST
1604 13EFE 8E80  RED236 GOSUBL STKVCT
           80
1605 13F04 56D          GONC   RED150
1606 13F07 7B27         GOSUB  GTPTRS
1607 13F0B 8AF          ?D#0   A
1608 13F0E 60           GOYES  RED238
1609 13F10 69D0         GOTO   EOFIL
1610 13F14 A47   RED238 D=D+D  S
1611 13F17 A47          D=D+D  S
1612 13F1A 560          GONC   RED240
1613 13F1D 6CB0         GOTO   REDASC
1614 13F21 A47   RED240 D=D+D  S
1615 13F24 460          GOC    RED41C
1616 13F27 6431         GOTO   REDFIX
1617             *
1618             * HP-41C DATA FILE
1619             *
1620 13F2B 870   RED41C ?ST=1  Notnum
1621 13F2E 83           GOYES  R41C30
1622 13F30 7FCC  R41C10 GOSUB  REDNUM
1623 13F34 865   R41C15 ?ST=0  Error
1624 13F37 60           GOYES  REDSTO
1625 13F39 6DE4  badtyp GOTO   BADTYP
1626             *
1627             * ASSIGN THE VALUE TO VARIABLE
1628             *
1629 13F3D 7BB6  REDSTO GOSUB  D1=stk        D1 @ MTHSTK
1630 13F41 1537  R41C25 A=DAT1 W             READ THE STACK TO A
1631 13F45 8E00         GOSUBL =STORE
           00
1632 13F4B 875          ?ST=1  Error
1633 13F4E 41           GOYES  R41CER
1634 13F50 78A6  R41C26 GOSUB  D1=stk
1635 13F54 8E00         GOSUBL =POPMTH
           00
1636 13F5A 779E         GOSUB  UPMTHS
1637 13F5E 669F         GOTO   RED230
1638             *
1639 13F62 6501  R41CER GOTO   RFXERR
1640             *
1641 13F66 871   R41C30 ?ST=1  Array
1642 13F69 F3           GOYES  R41C50
1643 13F6B 872          ?ST=1  String
1644 13F6E 81           GOYES  R41C40
1645             *
1646             * COMPLEX NUMBER
1647             *
1648 13F70 171   R41C35 D1=D1+ 2
1649 13F73 7C8C         GOSUB  REDNUM
1650 13F77 17F          D1=D1+ 16
1651 13F7A 758C         GOSUB  REDNUM
1652 13F7E 875          ?ST=1  Error
```

```
1653 13F81 8B          GOYES  badtyp
1654 13F83 59B         GONC   REDSTO
1655            *
1656            * SIMPLE STRING
1657            *
1658 13F86 171  R41C40 D1=D1+ 2            D1 @ STRING LENGTH
1659 13F89 704E        GOSUB  DROPST       D1 & STACK POINTER PAST THE STRING
1660 13F8D 1CF         D1=D1- 16
1661 13F90 7F6C R41C45 GOSUB  REDNUM
1662 13F94 17F         D1=D1+ 16
1663 13F97 865         ?ST=0  Error
1664 13F9A F9          GOYES  badtyp
1665 13F9C 8EA5        GOSUBL N-STR
          C0
1666 13FA2 845         ST=0   Error
1667 13FA5 5B9         GONC   R41C25       (B.E.T.)
1668            *
1669            * NUMERIC OR STRING ARRAY
1670            *
1671 13FA8 8EA3 R41C50 GOSUBL NXTADR       GET NEXT ELEMENT ADDRESS
          80
1672 13FAE 863         ?ST=0  Cmplex
1673 13FB1 80          GOYES  R41C55
1674 13FB3 1CF         D1=D1- 16
1675 13FB6 1C1         D1=D1- 2
1676 13FB9 76FD R41C55 GOSUB  STK16?
1677 13FBD 460         GOC    R41C60
1678 13FC0 6B04        GOTO   PRT185       INSUFFICIENT MEMORY
1679 13FC4 872  R41C60 ?ST=1  String
1680 13FC7 9C          GOYES  R41C45
1681 13FC9 782E        GOSUB  UPMTHS
1682 13FCD 77B2        GOSUB  LDE0         ASSUME IS COMPLEX NUMBER
1683 13FD1 873         ?ST=1  Cmplex
1684 13FD4 C9          GOYES  R41C35
1685 13FD6 695F        GOTO   R41C10
1686            *
1687            ********************
1688            * READING A TEXT FILE
1689 13FDA 870  REDASC ?ST=1  Notnum
1690 13FDD 04          GOYES  RASC30
1691 13FDF 7D0E        GOSUB  UPMH16       DROP 16 NIBS OF MTHSTK
1692 13FE3 748B RASC10 GOSUB  REDREC       READ THE NEXT RECORD FROM FILE
1693 13FE7 521         GONC   RASC20
1694 13FEA 71DA =EOFIL GOSUB  fpoll
1695 13FEE 52          CON(2) =pEOFIL
1696            *
1697            * End of file poll is fast poll. If is handled,
1698            *   don't ever come back here.
1699            * If ever come back here will be consided as not been handled.
1700            *
1701            *
1702 13FF0 3100 EOFILX LC(2)  =eEOFIL
1703 13FF4 8C00 ERREX  GOLONG =MfErr
          00
1704 13FFA 3451 RASC20 LC(5)  =STORTN
```

```
            041
 1705 14001 06          RSTK=C
 1706 14003 3400        LC(5)  =STORE
            000
 1707 1400A 06          RSTK=C
 1708               *
 1709 1400C 850         ST=1   =ValSub
 1710 1400F 8C00        GOLONG =VAL00
            00
 1711               *
 1712 14015 73ED =STORTN GOSUB  CKMFLG
 1713 14019 663F        GOTO   R41C26
 1714               *
 1715 1401D 871   RASC30 ?ST=1  Array
 1716 14020 82          GOYES  RASC50
 1717 14022 872         ?ST=1  String
 1718 14025 01          GOYES  RASC40
 1719               * COMPLEX NUMBER
 1720 14027 7FBD        GOSUB  UPMH34
 1721 1402B 7C3B  RASC35 GOSUB  REDREC        READ A RECORD FROM FILE
 1722 1402F 4AB         GOC    EOFIL
 1723 14032 57C         GONC   RASC20
 1724               *
 1725               * SIMPLE STRING
 1726               *
 1727 14035 719D  RASC40 GOSUB  DROP02
 1728 14039 7E2B  RASC45 GOSUB  REDREC
 1729 1403D 4CA         GOC    EOFIL
 1730 14040 71BD        GOSUB  UPMTHS
 1731 14044 6CFE        GOTO   R41C25
 1732               * NUMERIC/STRING ARRAY
 1733 14048 8EA9  RASC50 GOSUBL NXTADR        **Pack here...could save 2 nibs***
            70
 1734               *
 1735 1404E 873         ?ST=1  Cmplex
 1736 14051 AD          GOYES  RASC35
 1737 14053 872         ?ST=1  String
 1738 14056 3E          GOYES  RASC45
 1739 14058 6A8F        GOTO   RASC10
 1740               **
 1741               ** DATA FILE
 1742               **
 1743 1405C 870   REDFIX ?ST=1  Notnum
 1744 1405F 23          GOYES  RFX200
 1745               * SIMPLE NUMERIC VARIABLE
 1746 14061 7291  RFX040 GOSUB  DATBEG
 1747 14065 590         GONC   RFX050
 1748               *
 1749 14068 11B   RFXERR C=R3
 1750 1406B 688F        GOTO   ERREX
 1751               *
 1752 1406F 30A   RFX050 LCHEX  A
 1753 14072 982   RFX100 ?A<C   P             NEXT ITEM A NUMBER ?
 1754 14075 60          GOYES  RFX120
 1755 14077 6FA3  RFX110 GOTO   BADTYP
```

```
1756 1407B 748B  RFX120 GOSUB  REDNUM         READ THE NUMBER TO STACK
1757 1407F 5D0          GONC   RFX125
1758 14082 7E9C         GOSUB  EORC20
1759 14086 560          GONC   RFX125
1760 14089 6794         GOTO   RECERR
1761 1408D 6FAE  RFX125 GOTO   REDSTO
1762             *
1763 14091 871   RFX200 ?ST=1  Array
1764 14094 A2           GOYES  RFX220
1765 14096 872          ?ST=1  String
1766 14099 92           GOYES  RFX250
1767             * COMPLEX NUMBER
1768 1409B 7851  RFX210 GOSUB  DATBEG
1769 1409F 48C          GOC    RFXERR
1770 140A2 30A          LCHEX  A
1771 140A5 98E          ?A>=C  P               IS NEXT ITEM A NUMBER ?
1772 140A8 FC           GOYES  RFX110          IF NOT, BAD TYPE
1773 140AA 171          D1=D1+ 2
1774 140AD 725B         GOSUB  REDNUM
1775 140B1 17F          D1=D1+ 16
1776             **********************************
1777             * Pack here could save 1 nibble
1778             *
1779             *      GOSUB  LDC=8
1780             *
1781 140B4 D2           C=0    A
1782 140B6 308          LC(1)  8
1783             *
1784             * End of possible packing
1785             ****************************************
1786 140B9 E1           B=B-C  A               B= BYTES LEFT IN CUR.RECORD
1787 140BB 55A          GONC   RFX040          (B.E.T.)
1788 140BE 6AF0  RFX220 GOTO   RFXARY
1789             *
1790             * SIMPLE STRING
1791             *
1792 140C2 17B   RFX250 D1=D1+ 12              D1 @ MAX. STRING LENGTH
1793 140C5 D2           C=0    A
1794 140C7 15F3         C=DAT1 4
1795 140CB 10B          R3=C                   SAVE MAX. LENGTH IN R3
1796 140CE 1C9          D1=D1- 10              D1@ STRING LENGTH IN NIBS
1797 140D1 78FC         GOSUB  DROPST
1798 140D5 858   RFX260 ST=1   Fresh
1799 140D8 7B11  RFX270 GOSUB  DATBEG          SKIP TO NEXT DATA ITEM BEGIN
1800 140DC 5B0          GONC   RFX280
1801 140DF 878          ?ST=1  Fresh
1802 140E2 68           GOYES  RFXERR
1803 140E4 6FB0         GOTO   RFX390          (Changed from GOC RFX300 8/29/83)
1804             *
1805 140E8 31FC  RFX280 LCHEX  CF
1806 140EC 902          ?A=C   P               IS NEXT ITEM A STRING ?
1807 140EF D0           GOYES  RFX310
1808 140F1 878   RFX290 ?ST=1  Fresh
1809 140F4 38           GOYES  RFX110
1810 140F6 20           P=     0               (Changed from ST=1 Error 8/29/83)
```

```
1811 140F8 60A0          GOTO   RFX382
1812 140FC 878   RFX310  ?ST=1  Fresh
1813 140FF 70            GOYES  RFX320
1814 14101 9EE           ?A>=C  B
1815 14104 DE            GOYES  RFX290
1816 14106 7829  RFX320  GOSUB  D0+2RD        POINTS TO STRING LENGTH
1817 1410A 79F8          GOSUB  RDLNFX        READ STRING LENGTH
1818 1410E 11B           C=R3                 C= MAX. STRING LENGTH
1819 14111 8BA           ?A<=C  A
1820 14114 E0            GOYES  RFX330
1821 14116 7EE9          GOSUB  BACK3B
1822 1411A 3100 =STROVF  LC(2)  =eSTROV       SAY STRING OVER FLOW
1823 1411E 65DE          GOTO   ERREX
1824             *
1825 14122 D2    RFX330  C=0    A             TEST IF THIS IS LAST PART OF
1826 14124 303           LC(1)  3              THE STRING
1827 14127 E1            B=B-C  A             B= #OF BYTES IN CURRENT RECORD
1828 14129 846           ST=0   Done           CAN BE USED TO STORE DATA
1829 1412C 8B0           ?B<A   A             CAN REST OF STR. IN THIS RECORD?
1830 1412F 70            GOYES  RFX340
1831 14131 856           ST=1   Done          IF SO, SET DONE FLAG
1832 14134 D8            B=A    A             B= # OF DATA BYTE IN THIS RECORD
1833             *
1834 14136 C9    RFX340  C=C+B  A             C= # OF BYTES TO PAST THIS TIME
1835 14138 E3            D=D-C  A             SEE IF PAST END OF FILE
1836 1413A 5A0           GONC   RFX350
1837 1413D 77C9          GOSUB  BACK3B
1838 14141 68AE          GOTO   EOFIL
1839             *
1840 14145 137   RFX350  CD1EX
1841 14148 06            RSTK=C               SAVE CURRENT STACK POINTER
1842 1414A 816           CSRC   W
1843 1414D CE            C=C-1  A             16 NIBBLES FOR STRING HEADER
1844 1414F 812           CSLC   W
1845 14152 AFA           A=C    W
1846 14155 E0            A=A-B  A
1847 14157 E0            A=A-B  A
1848 14159 7C5C          GOSUB  STKSPC        MAKE SURE STILL ROOM ON STACK
1849 1415D 07            C=RSTK
1850 1415F 135           D1=C
1851 14162 4A0           GOC    RFX360
1852 14165 7F99          GOSUB  BACK3B
1853             *
1854 14169 6262  RFX355  GOTO   PRT185
1855 1416D 848   RFX360  ST=0   Fresh
1856 14170 CD    RFX370  B=B-1  A             MOVE THE STRING TO STACK
1857 14172 431           GOC    RFX380
1858 14175 14A           A=DAT0 B
1859 14178 1C1           D1=D1- 2
1860 1417B 149           DAT1=A B
1861 1417E 70B8          GOSUB  D0+2RD
1862 14182 6DEF          GOTO   RFX370
1863             *
1864 14186 7DDA  RFX380  GOSUB  UPCPOS
1865 1418A 876           ?ST=1  Done
```

```
1866 1418D 71          GOYES  RFX390
1867 1418F 119         C=R1
1868 14192 D5          B=C    A              B= RECORD LENGTH
1869 14194 869         ?ST=0  Random         SERIAL ACCESS ?
1870 14197 90          GOYES  RFX385
1871 14199 7D80  RFX382 GOSUB  DATBER
1872 1419D 460         GOC    RFX390         (B.E.T.)
1873 141A0 673F  RFX385 GOTO   RFX270
1874               *
1875               * DONE WITH READING THE STRING FROM FILE TO STACK
1876               * NOW COMPUTE TOTAL STRING LENGTH ON STACK
1877               *
1878 141A4 8E48  RFX390 GOSUBL STRHED
           A0
1879 141AA 7E4B        GOSUB  EORCHK
1880 141AE 560         GONC   RFX395
1881 141B1 7570        GOSUB  DATBER
1882 141B5 678D  RFX395 GOTO   REDSTO
1883               *
1884               * NUMERIC/ STRING ARRAY
1885               *
1886 141B9 7B26  RFXARY GOSUB  NXTADR
1887 141BD 862         ?ST=0  String
1888 141C0 60          GOYES  RFX420
1889 141C2 621F        GOTO   RFX260
1890 141C6 863   RFX420 ?ST=0  Cmplex
1891 141C9 80          GOYES  RFX430
1892 141CB 1CF         D1=D1- 16
1893 141CE 1C1         D1=D1- 2
1894 141D1 7EDB  RFX430 GOSUB  STK16?
1895 141D5 571         GONC   RFX440
1896 141D8 791C        GOSUB  UPMTHS
1897 141DC 863         ?ST=0  Cmplex
1898 141DF A0          GOYES  RFX435
1899 141E1 73A0        GOSUB  LDE0
1900 141E5 65BE        GOTO   RFX210
1901 141E9 677E  RFX435 GOTO   RFX040
1902               *
1903 141ED 6ED1  RFX440 GOTO   PRT185
1904               *
1905               *
1906 141F1 8C00  saved0 GOLONG =SAVED0
           00
1907               *
1908               ********************************************************************
1909               ********************************************************************
1910               **
1911               ** Name:    DATBEG  -  Detect End of Record of DATA File
1912               **
1913               ** Category:   FILUTL
1914               **
1915               ** Purpose:  Detect end of record in a DATA file
1916               **
1917               ** Entry: R0(A) = Address of next byte to read
1918               **        D = # of bytes to end of file
```

```
1919            **        S9 = 0/1 for serial/random access
1920            **        B = # of bytes left in current record.
1921            ** Exit:
1922            **        Carry clear => No error
1923            **         D0 @ Next data byte to read
1924            **        Carry set => Reached end of file or end of record
1925            **                     in random access.
1926            **         C(3-0) & R3 = Error number
1927            **         S6=1
1928            ** Uses A, C, D0
1929            ********************************************************************
1930            ********************************************************************
1931            *
1932 141F7 110   DATBEG A=R0
1933 141FA 130          D0=A
1934 141FD 14A   DATB05 A=DAT0 B
1935 14200 31FF         LC(2)  =fEOF
1936 14204 966          ?A#C   B
1937 14207 31           GOYES  DATB20
1938 14209 869          ?ST=0  Random
1939 1420C 62           GOYES  DATB30
1940 1420E 3100  DATB10 LC(2)  =eEOFIL
1941 14212 10B   DATB15 R3=C
1942 14215 855          ST=1   Error
1943 14218 02           RTNSC
1944            *
1945 1421A 31FE  DATB20 LC(2)  =fEOR
1946 1421E 962          ?A=C   B              REACHED END OF RECORD ?
1947 14221 40           GOYES  DATB25         IF SO, SEE IF RANDOM ACCESS
1948 14223 03           RTNCC
1949            *
1950            *
1951 14225 869   DATB25 ?ST=0  Random         SERIAL ACCESS ?
1952 14228 A0           GOYES  DATB30         IF SO, SEE IF MORE THAN 8 BYTES LEF
1953            *
1954 1422A 3100  DATBER LC(2)  =eRECOR
1955 1422E 63EF         GOTO   DATB15
1956            *
1957            * In serial access if the file pointer is pointing at an EOF or E
1958            * and is at beginning of a record, give "End of File" error.
1959            * If not at beginning of a record, skip to look at next record.
1960            *
1961 14232 119   DATB30 C=R1
1962 14235 8B9          ?B>=C  A              At the beginning of the record ?
1963 14238 6D           GOYES  DATB10         If so, say "End of File"
1964 1423A D9           C=B    A              C= BYTES LEFT IN CUR. RECORD
1965 1423C E3           D=D-C  A              D= BYTES TO END OF FILE
1966 1423E 4FC          GOC    DATB10
1967 14241 8AB          ?D=0   A
1968 14244 AC           GOYES  DATB10
1969            *
1970 14246 D4           A=B    A              A(A) = # of bytes left in current
1971            *                             record.
1972 14248 CC    DATB55 A=A-1  A
1973 1424A 4C0          GOC    DATB60
```

```
1974 1424D 8EFD          GOSUBL DO+2RD
           7F
1975 14253 64FF          GOTO   DATB55
1976              *
1977 14257 118   DATB60  C=RO
1978 1425A 136           CDOEX
1979 1425D 108           RO=C
1980 14260 111           A=R1
1981 14263 D8            B=A    A              B= RECORD LENGTH
1982 14265 619F          GOTO   DATBEG         START IT OVER AGAIN
1983              *
1984              **
1985 14269        FXSP90
1986 14269 E3            D=D-C  A
1987 1426B 112           A=R2
1988 1426E D8            B=A    A
1989 14270 01            RTN
1990              *
1991 14272 D2    LDC=8   C=0    A
1992 14274 308           LC(1)  8
1993 14277 01            RTN
1994              *
1995              *
1996 14279 0             CON(1) =FIXSPC        *** THERE IS 12 NIBBLES HERE ***
1997 1427A               BSS    12-1
1998              *
1999              *
2000 14285 1C1   LDEO-   D1=D1- 2
2001 14288 31EO  LDEO    LCHEX  OE
2002 1428C 14D           DAT1=C B
2003 1428F 03            RTNCC
2004              *
2005              *
2006              ********************************************************************
2007              ********************************************************************
2008              ** Name:    PRINT#  -  PRINT# Statement Execute
2009              **
2010              ** Category: STEXEC
2011              **
2012              ** Purpose:  Write data to data file.
2013              **           The code here works for file both in RAM or mass
2014              **           memory device. But will only work with type TEXT,
2015              **           SDATA and DATA files. It will poll for unrecognized
2016              **           file type.
2017              **
2018              ** Entry: DO points past the PRINT# token
2019              **
2020              ** Exit: Exit to NXTSTM
2021              **
2022              ** Detail:
2023              **  . PRINT# only works with an opened file. When a file is
2024              **    opened, an entry in FIB is created. The FIB keep track all
2025              **    information relate to that file, such as file type, file
2026              **    is a memory resident file or external file, current file
2027              **    pointer.
```

```
2028            **
2029            **  . If the file is in an external device, there should be an
2030            **    I/O buffer containing the current sector of the file. The
2031            **    PRINT# writes data direct to the I/O buffer. When the last
2032            **    byte of the buffer is filled, the PRINT# will poll the
2033            **    HPIL ROM to write the I/O buffer back to the file and read
2034            **    the next sector into the I/O buffer.
2035            **
2036            **  . If the file copy code is 8, the PRINT# will issue a poll
2037            **    to see if there is any LEX file want to handle it. The LEX
2038            **    file is supposed to take over the PRINT# statement from
2039            **    there.
2040            **    But the PRINT# execution will do the following things
2041            **    before pulling for copy code 8 file :
2042            **    1. Get the channel number and find the FIB entry address
2043            **       of the file. The FIB entry address is stored in STMTD1.
2044            **    2. Get the record number, if specified, and set the file
2045            **       pointer to the start of the specified record. Since the
2046            **       mainframe code doesn't know how to search a record in
2047            **       the copy code 8 file, a poll (pSREC#) will be issued
2048            **       for that purpose.
2049            ********************************************************************
2050            ********************************************************************
2051            *
2052            ********************************************************************
2053            ********************************************************************
2054            **
2055            ** Name:(S) pPRIN#  -  PRINT# on File of Copycode = 8
2056            **
2057            ** Category:   POLL
2058            **
2059            ** Type:       POLL
2060            **
2061            **
2062            ** Purpose:
2063            **      Execution of PRINT # statement when the copy code of the
2064            **      file is 8.
2065            **
2066            ** Should poll be "Handled" (return with XM=0)?:Yes
2067            **
2068            ** Meaning of "Handling" Poll (what does code do if handled?):
2069            **      Complete the execution of PRINT # statement.
2070            **
2071            ** Entry conditions for handler (registers, ST, RAM, etc.):
2072            **      B[A] = Poll number.
2073            **      HEX mode.
2074            **      P=0.
2075            **
2076            **      D[S] = Copy code of the file.
2077            **      D(A) = # of bytes to end of file
2078            **      R0(A)= Current position (absolute address)
2079            **      R0(15:14) = Relative position in buffer if external
2080            **      R1   = Record length in bytes
2081            **      CHN#SV = Channel # specified in the statemnt.
2082            **      STMTD1 = FIB entry address of the file.
```

```
2083            **       (All the file related information can be found in the
2084            **        FIB entry of the file)
2085            **      STMTDO = Program counter points at the semmicolon of the
2086            **               statement.
2087            **      S9  = 0 if serial access (record # not specified)
2088            **          = 1 If random access
2089            **      S10 = 0 if file is in mainframe RAM/ROM
2090            **          = 1 if file is in external mass memory deivce
2091            **      S11 = 0 if file is not in Independent RAM
2092            **          = 1 if file is in Independent RAM
2093            **
2094            **      At the time when this poll is issued, the READ#
2095            **      already process the channel number and the record number
2096            **      -if specified.
2097            **      If the record number has been specified, the pSREC# poll
2098            **      should been issued earlier so the file pointer(in the
2099            **      FIB) should already pointing at the start of the record.
2100            **
2101            **
2102            ** Normal exit conditions from handler if handled (ST, RAM,
2103            ** registers, etc.):
2104            **      If handled, the handler should complete the PRINT#
2105            **      statement and directly exit to NXTSTM. The math stack
2106            **      will be cleared by the run loop automatically.
2107            **
2108            ** Normal exit conditions from handler if not handled (ST, RAM,
2109            ** registers, etc.):
2110            **      Carry clear.
2111            **      HEX mode.
2112            **      XM=1.
2113            **
2114            ** Error exit conditions from handler:
2115            **      Carry set.
2116            **      HEX mode.
2117            **      C[0-3] = Error number.
2118            **
2119            ** Available subroutine levels: 6
2120            **
2121            ** What registers/RAM may be used if handled?:
2122            **      All CPU registers, scratch RAM, ST11-0
2123            **
2124            ** What registers/RAM may be used if not handled?:
2125            **      B,C, D0, D1
2126            **
2127            ** What registers/RAM may be used if error exit ?:
2128            **      All CPU registers, scratch RAM, ST11-0
2129            **
2130            ** History:
2131            **
2132            **    Date     Programmer              Modification
2133            **  --------   ----------   ------------------------------------
2134            **  04/20/83   SC           Document
2135            **
2136            ********************************************************************
2137            ********************************************************************
```

```
2138                *
2139 14291 161     =PRINT# D0=D0+ 2
2140 14294 8E00             GOSUBL =GETCH#       GET CHANNEL #
           00
2141 1429A 8E00             GOSUBL =GETRE#       GET RECORD #
           00
2142                * CHANNEL # IS SAVED IN CHN#SV
2143                *
2144 142A0 7FDA             GOSUB  CHKCOD        SEE WHAT IS THE FILE TYPE
2145 142A4 512              GONC   PRT110
2146                *
2147 142A7 718B    PRTPOL   GOSUB  poll
2148 142AB 62               CON(2) =pPRIN#
2149 142AD 590     PRT100   GONC   PRT105
2150                *
2151                * Unknown data type handler comes back with error
2152                *
2153 142B0 8D00    =BsErr   GOVLNG =BSERR         Display the Lex file error message
           000
2154                *
2155 142B7 831     PRT105   ?XM=0
2156 142BA 80               GOYES  PRT106         HAS BEEN HANDLED
2157 142BC 8C00    ftyper   GOLONG =FTYPER
           00
2158 142C2 61C0    PRT106   GOTO   PRT140
2159                *
2160 142C6 8E9F    PRT110   GOSUBL D0=FIB
           7F
2161 142CC 168              D0=D0+ (oPROTb)
2162 142CF 1520             A=DAT0 P
2163 142D3 301              LC(1)  1
2164 142D6 0E06             A=A&C  P
2165 142DA 908              ?A=0   P              FILE PROTECTED ?
2166 142DD 90               GOYES  PRT120         IF NOT, GO TO PRT120
2167 142DF 3100             LC(2)  =eFPROT
2168 142E3 591              GONC   PRTERR         (B.E.T.)
2169                *
2170 142E6 816     PRT120   CSRC
2171                * PRINT # CAN ONLY PRINT TO MAINFRAME RAM/IRAM AND EXTERNAL FILE
2172 142E9 949              ?B=0   S
2173 142EC 51               GOYES  PRT121         0 IS RAM
2174 142EE 941              ?B=C   S
2175 142F1 01               GOYES  PRT121         1 IS IRAM
2176 142F3 A45              B=B+B  S
2177 142F6 4A0              GOC    PRT121         8 IS EXTERNAL FILE
2178 142F9 3100             LC(2)  =eFACCS
2179 142FD 66FC    PRTERR   GOTO   ERREX
2180 14301 8E1F    PRT121   GOSUBL SETWRT         SET ACCESS MODE TO WRITE
           7F
2181 14307 7C5A             GOSUB  EOLCHK
2182 1430B 560              GONC   PRT123         AT END OF FILE
2183 1430E 63C0             GOTO   PRT200         NOT AT END OF LINE YET
2184                * Here is the PRINT # statement with no print list
2185                * If the file is LIF ASCII file, write an end of file mark(FFFF).
2186                * If the file is the fixed length data file:
```

```
2187                *    Write end of data mark(EF) if is a random print
2188                *    Write end of file mark(FF) if is a serial print
2189                *
2190 14312 7023  PRT123 GOSUB  GTPTRS
2191 14316 845         ST=0   Error
2192 14319 854         ST=1   Return
2193 1431C D2          C=0    A
2194 1431E E6          C=C+1  A
2195 14320 A4F         D=D-1  S
2196 14323 A4F         D=D-1  S
2197 14326 562         GONC   PRT130
2198                *
2199                * DATA FILE
2200                *
2201 14329 7B37  PRT124 GOSUB  GETSPC         SEE IF STILL ROOM IN FILE
2202 1432D 443         GOC    PRT135         IF NOT, ERROR
2203 14330 31FF        LC(2)  =fEOF
2204 14334 869         ?ST=0  Random         SERIAL ACCESS ?
2205 14337 60          GOYES  PRT125
2206 14339 31FE        LC(2)  =fEOR
2207 1433D 8E02  PRT125 GOSUBL WRTMRK
           7F
2208 14343 8E3C        GOSUBL BACK1B
           7F
2209 14349 6670        GOTO   PRT180
2210                *
2211 1434D A4F   PRT130 D=D-1  S
2212 14350 433         GOC    PRT140
2213                *
2214                * TEXT FILE
2215                *
2216 14353 869         ?ST=0  Random         Only allow serial access
2217 14356 60          GOYES  PRT131
2218 14358 636F        GOTO   ftyper
2219                *
2220 1435C C6    PRT131 C=C+C  A              C(A) = 2
2221 1435E 7607        GOSUB  GETSPC         GET SPACE FOR 2 BYTES
2222 14362 5B0   PRT135 GONC   PRT136
2223 14365 86A         ?ST=0  I/Obuf
2224 14368 46          GOYES  PRT185
2225 1436A 658C        GOTO   EOFILX
2226                *
2227 1436E 31FF  PRT136 LC(2)  =fEOF
2228 14372 8EBE        GOSUBL WRTMRK
           6F
2229 14378 8EDD        GOSUBL WRTEOF
           6F
2230 1437E 8E68        GOSUBL BACK2B
           7F
2231                *
2232 14384         =NXtstm
2233 14384 8C00  PRT140 GOLONG =NEXTst        To NXTSTM
           00
2234                *
2235                *
```

```
2236 1438A 79D9  PRT150 GOSUB  EOLCHK
2237 1438E 434          GOC    PRT200
2238             *
2239             * DONE WITH PRINT STATEMENT
2240             *  IF THIS IS A SERIES PRINT, WRITE AN END OF FILE MARK TO THE
2241             *  CURRENT POSITION
2242             *
2243             *
2244             * Bug fix by NZ on 8/18/83 to fix the bug that does not rewrite
2245             * the end-of-file mark following a random print to the last
2246             * record in the file (the old EOF mark)
2247             *
2248             *
2249             * The modified code checks if this is a Random PRINT which
2250             * exactly filled the LAST logical record in the file.  If it is,
2251             * then an End-of-File mark needs to be written here, even though
2252             * it will be in the next record, to make sure that DATA files in
2253             * RAM always have an EOF.  (NZ)
2254             *
2255             *PRT170 ?ST=1  Random         (Removed by NZ)
2256             *       GOYES  PRT180         (Removed by NZ)
2257             *
2258 14391 71A2  PRT170 GOSUB  GTPTRS
2259 14395 A4F          D=D-1  S              IF NOT THE FIXED LENGTH FILE
2260 14398 A4F          D=D-1  S              DON'T WRITE EOF
2261 1439B 542          GONC   PRT180
2262 1439E 854          ST=1   Return
2263 143A1 D2           C=0    A
2264 143A3 E6           C=C+1  A
2265 143A5 8AF          ?D#0   A              AT END OF FILE ?
2266 143A8 70           GOYES  PRT175         IF NOT, WRITE IT
2267             * Don't write end of file mark if reached end of an external
2268             * file.
2269             *
2270 143AA 87A          ?ST=1  I/Obuf
2271 143AD 31           GOYES  PRT180
2272             *
2273 143AF       PRT175
2274             *
2275             * New lines to fix bug (See comments above)
2276             *
2277 143AF 869          ?ST=0  Random         Is this a random print?
2278 143B2 A0           GOYES  PRT178         No...skip this fix code
2279 143B4 8A7          ?C#D   A              Exactly 1 byte left in file?
2280 143B7 90           GOYES  PRT180         No...don't write anything
2281 143B9 849          ST=0   Random         Fake PRT124 to write EOF, not EOR
2282 143BC       PRT178
2283             *
2284             * End of bug fix code
2285             *
2286 143BC 6C6F         GOTO   PRT124         WRITE EOF
2287             *
2288 143C0 865   PRT180 ?ST=0  Error
2289 143C3 1C           GOYES  PRT140
2290 143C5 11B          C=R3
```

```
2291 143C8 6B2C         GOTO   ERREX
2292 143CC 8C00  PRT185 GOLONG =NOMEM
           00
2293            *
2294 143D2 161   PRT200 DO=DO+ 2
2295 143D5 8E00         GOSUBL =EXPEX-        GET THE NEXT PRINT LIST TO STACK
           00
2296 143DB 721E         GOSUB  savedO         SAVE DO IN STMTDO
2297 143DF 31F8         LCHEX  8F             CHANGE 8F TO 0F
2298 143E3 966          ?A#C   B
2299 143E6 90           GOYES  PRT201
2300 143E8 31F0         LCHEX  0F
2301 143EC 14D          DAT1=C B
2302 143EF 790A  PRT201 GOSUB  CKMFLG
2303 143F3 6010         GOTO   PRT215
2304            *
2305 143F7 773A  LOOPBK GOSUB  CHKDON
2306 143FB 560          GONC   PRT210
2307 143FE 6B8F  NXTLST GOTO   PRT150
2308            *
2309 14402 08    PRT210 CLRST
2310 14404 7403  PRT215 GOSUB  STKVCT         CHECK DATA TYPE ON THE STACK
2311 14408 55F          GONC   NXTLST
2312 1440B 7722  PRT220 GOSUB  GTPTRS         GET POINTERS OF THE FILE FROM FIB
2313 1440F A47          D=D+D  S
2314 14412 A47          D=D+D  S
2315            ********************************
2316            * Pack by NZ on 8/18/83...saves 4 nibbles
2317            *
2318 14415 4E6          GOC    ASCIIF         This is an ASCII file
2319            *
2320            *       GONC   PRT230
2321            *       GOTO   ASCIIF         IS A ASCII FILE
2322            *
2323            * End of pack by NZ
2324            ********************************
2325 14418 A47   PRT230 D=D+D  S
2326 1441B 460          GOC    HP41DT
2327 1441E 62E0         GOTO   FIXLEN         HP-71 FIXED LENGTH FILE
2328            ************************
2329            * SDATA FILE
2330            *
2331 14422 862   HP41DT ?ST=0  String         STRING ?
2332 14425 80           GOYES  H41D10         DON'T DO STRING ON HP-41C FILE
2333 14427 8C00  BADTYP GOLONG =RDATTY           SAY "DATA TYPE"
           00
2334            *
2335            ***********************************
2336            * Pack here could save 1 nibble
2337            *       GOSUB   LDC=8
2338            *
2339 1442D D2    H41D10 C=0     A
2340 1442F 308          LC(1)  8
2341            *
2342            * End of possible packing
```

```
2343            **********************************
2344 14432 870          ?ST=1  Notnum
2345 14435 C2           GOYES  H41D40          NOT A NUMBER
2346            * WRITE A SIMPLE NUMBER TO HP-41C DATA FILE
2347 14437 7D26         GOSUB  GETSPC          MAKE SURE THERE IS ENOUGH ROOM
2348 1443B 511          GONC   H41D20          ENOUGH ROOM
2349 1443E 855   EOFERX ST=1   Error
2350 14441 3100         LC(2)  =eEOFIL
2351 14445 86A          ?ST=0  I/Obuf          Internal file ?
2352 14448 13           GOYES  NOSPC           If so, say "Insufficient Memory"
2353            *
2354 1444A 523          GONC   NOSPC2
2355            *
2356 1444D 110   H41D20 A=R0
2357 14450 130          D0=A                   D0 @ CURRENT FILE POINTER
2358 14453 1537         A=DAT1 W
2359 14457 8E76         GOSUBL WRTNUM
           5F
2360 1445D 699F         GOTO   LOOPBK
2361            *
2362 14461 871   H41D40 ?ST=1  Array
2363 14464 C0           GOYES  H41D50
2364            *
2365            * MUST BE A COMPLEX NUMBER
2366            *
2367 14466 171          D1=D1+ 2               Step over the E0
2368 14469 7889  H41D45 GOSUB  UPMTHS
2369 1446D 549          GONC   PRT210          (B.E.T.)
2370            *
2371            * NUMERIC ARRAY
2372            *
2373 14470 7834  H41D50 GOSUB  NXTELM
2374 14474 865          ?ST=0  Error
2375 14477 B8           GOYES  PRT210
2376            *
2377 14479 3100  NOSPC  LC(2)  =eMEM
2378 1447D 10B   NOSPC2 R3=C
2379 14480 601F         GOTO   PRT170
2380            **********************
2381            * TEXT FILE
2382            *
2383 14484 869   ASCIIF ?ST=0  Random      Only allow serial print to text file
2384 14487 60           GOYES  ASCSEC
2385 14489 623E         GOTO   ftyper        Say "Invalid File Type"
2386            *
2387 1448D 870   ASCSEC ?ST=1  Notnum
2388 14490 C0           GOYES  ASC100
2389            * SIMPLE NUMBER ON STACK
2390 14492 8E00  ASCN50 GOSUBL =STR$SB        CONVERT THE NUMBER TO ASCII
           00
2391 14498 60DF         GOTO   H41D45          UPDATE MATH STACK POINTER
2392 1449C 871   ASC100 ?ST=1  Array
2393 1449F 81           GOYES  ASARRY
2394 144A1 862          ?ST=0  String
2395 144A4 EE           GOYES  ASCN50          MUST BE A SIMPLE COMPLEX NUMBER
```

```
2396                *
2397                * SIMPLE STRING ON STACK
2398                *
2399 144A6 7A95  ASCSTR GOSUB  STRSPC          SEE IF THERE IS ROOM IN THE FILE
2400 144AA 439   ASC230 GOC    EOFERX
2401 144AD 8ECB         GOSUBL WRTSTR          WRITE THE STRING TO FILE
           4F
2402 144B3 634F         GOTO   LOOPBK          SEE IF DONE WITH ONE ITEM
2403                *
2404                * NUMERIC OR STRING ARRAY
2405                *
2406 144B7 872   ASARRY ?ST=1  String
2407 144BA 71           GOYES  STRARY
2408                *
2409                * NUMERIC ARRAY
2410                *
2411 144BC 7CE3         GOSUB  NXTELM
2412 144C0 875          ?ST=1  Error           ENOUGH ROOM ON STACK ?
2413 144C3 6B           GOYES  NOSPC
2414 144C5 863          ?ST=0  Cmplex
2415 144C8 AC           GOYES  ASCN50
2416 144CA 77BD         GOSUB  LDE0-
2417 144CE 53C          GONC   ASCN50          CONVERT THE NUMBER TO ASCII
2418                * STRING ARRAY
2419 144D1 857   STRARY ST=1   Check
2420 144D4 74D3         GOSUB  NXTELM
2421 144D8 7475         GOSUB  STSPC-          SEE IF STILL ROOM IN FILE
2422 144DC 4DC   ASC240 GOC    ASC230          NOT ENOUGH ROOM
2423 144DF 847          ST=0   Check
2424 144E2 76C3         GOSUB  NXTELM
2425 144E6 AF2          C=0    W
2426 144E9 15E3         C=DATO 4               C= STRING LENGTH
2427 144ED 163          D0=D0+ 4
2428 144F0 C6           C=C+C  A               STRING LENGTH IN NIBBLES
2429 144F2 132          ADOEX
2430 144F5 CA           A=A+C  A
2431 144F7 8E67         GOSUBL WRSTR*          WRITE THE STRING TO FILE
           4F
2432 144FD 69FE         GOTO   LOOPBK
2433                **************
2434                * HP-71 DATA FILE
2435                *
2436 14501 860   FIXLEN ?ST=0  Notnum
2437 14504 60           GOYES  FIX100
2438 14506 6380         GOTO   FIX300
2439                * B(A) = # OF BYTES LEFT IN CURRENT RECORD(THESE BYTES MAY NOT
2440                *        EXIST IN THE FILE YET)
2441                ************************************
2442                * Pack here could save 1 nibble
2443                *
2444                * FIX100 GOSUB  LDC=8
2445                *
2446 1450A D2    FIX100 C=0    A               B= BYTES LEFT IN CURRENT RECORD
2447 1450C 308          LC(1)  8
2448                *
```

```
2449                * End of possible packing
2450                **********************************
2451 1450F 8B9            ?B>=C  A              More than 8 bytes left in record?
2452 14512 91             GOYES  FIX110         Yes...we only need 8 (continue)
2453                *
2454 14514 879            ?ST=1  Random         IS THIS A RANDOM ACCESS ?
2455 14517 A0             GOYES  RECERR
2456 14519 111            A=R1                  A(A) = RECORD LENGTH
2457 1451C 8BE            ?A>=C  A              RECORD LENGTH >= 8 ?
2458 1451F A0             GOYES  FIX105
2459 14521 3100  RECERR   LC(2)  =eRECOR        SAY REACHED END OF RECORD
2460 14525 6ECA           GOTO   ERREX
2461                *
2462 14529 C9     FIX105  C=C+B  A              TOTAL # OF BYTES NEEDED
2463 1452B 87A    FIX110  ?ST=1  I/Obuf         EXTERNAL FILE ?
2464 1452E 40             GOYES  FIX115
2465 14530 E6             C=C+1  A              INTERNAL FILE NEED ONE MORE BYTE
2466 14532 7235  FIX115   GOSUB  GETSPC         SEE IF STILL ROOM AVAILABLE ?
2467 14536 45A   FIX118   GOC    ASC240         IF NOT, ERROR OUT
2468 14539 110            A=R0
2469 1453C 130            DO=A                  DO @ CURRENT FILE PTR
2470 1453F 7F2D           GOSUB  LDC=8          C(A) = 00008
2471 14543 8B9            ?B>=C  A              Can this fit in current record?
2472 14546 B1             GOYES  FIX145         Yes...go write the number
2473 14548 111            A=R1                  A= RECORD LENGTH
2474 1454B DC             ABEX   A              A=# OF BYTES LEFT IN CURRENT REC.
2475 1454D E1             B=B-C  A              B=# OF BYTES LEFT IN NEXT RECORD
2476 1454F CC     FIX125  A=A-1  A              SKIP OVER CURRENT RECORD
2477 14551 4F0            GOC    FIX145
2478 14554 8EC1           GOSUBL DO+2WR
             5F
2479 1455A 64FF           GOTO   FIX125         SEE IF ENOUGH ROOM
2480                *
2481 1455E 47D   FIX140   GOC    FIX118         NOT ENOUGH ROOM TO EXPAND
2482 14561 1537  FIX145   A=DAT1 W              READ THE NUMBER AGAIN
2483 14565 8E95           GOSUBL WRTNUM         WRITE THE NUMBER TO FILE
             4F
2484 1456B 421            GOC    FIX160
2485 1456E 8EDE           GOSUBL WSTF80         WRITE END OF RECORD MARK
             3F
2486 14574 8EE7           GOSUBL SETWRT         PUT BACK TO WRITE MODE
             5F
2487 1457A 6C7E  FIX150   GOTO   LOOPBK
2488                *
2489 1457E 8E0A  FIX160   GOSUBL EORC20         UPDATE CURRENT POSITION IN FIB
             7F
2490 14584 55F   FIX163   GONC   FIX150
2491 14587 499   FIX165   GOC    RECERR
2492                *
2493                * NOT A SIMPLE NUMBER
2494                *
2495 1458A 871   FIX300   ?ST=1  Array
2496 1458D 21             GOYES  FIXARY
2497 1458F 872            ?ST=1  String
2498 14592 61             GOYES  FIXSTR
```

```
2499                *
2500                * COMPLEX NUMBER, JUST TREAT IT AS TWO NUMBER
2501                *
2502 14594 171             D1=D1+ 2              Step over the EO
2503 14597 7A58            GOSUB  UPMTHS
2504 1459B 666E            GOTO   PRT210
2505                *
2506 1459F 872    FIXARY ?ST=1  String
2507 145A2 83              GOYES  FXSTAR
2508                *
2509                * NUMERIC ARRAY
2510                *
2511 145A4 6BCE            GOTO   H41D50         GET NEXT ELEMENT OF ARRAY
2512                *
2513                * SIMPLE STRING
2514                * B(A) = # OF BYTES LEFT IN CURRENT RECORD
2515                * D = # OF BYTES TO END OF FILE
2516                * D1 @ TOP OF STACK
2517 145A8 171    FIXSTR D1=D1+ 2
2518 145AB AF0             A=0    W
2519 145AE 143             A=DAT1 A              A= STRING LENGTH IN NIBBLES
2520 145B1 81C             ASRB                 A= STRING LENGTH IN BYTES
2521 145B4 7A24            GOSUB  FXSTSP         COMPUTE TOTAL SPACE NEEDED
2522 145B8 45A    FIX350   GOC    FIX140         NOT ENOUGH ROOM
2523 145BB 8EC0            GOSUBL DROPST         POINTS TO END OF STRING
           8F
2524 145C1 AFA             A=C    W
2525 145C4 81C             ASRB                 A= STRING LENGTH IN BYTES
2526 145C7 8E8E   FIX360   GOSUBL WSTRFX         WRITE STRING TO FILE
           2F
2527 145CD 49B             GOC    FIX165
2528                *
2529                * UPDATE CURRENT POSITION AND CHECK CROSSING END OF RECORD
2530                *
2531 145D0 8E62            GOSUBL EORCHK
           7F
2532                *
2533 145D6 6DAF            GOTO   FIX163         Check if end of record (carry=end)
2534                *
2535                * STRING ARRAY
2536 145DA 857    FXSTAR ST=1   Check
2537 145DD 7BC2            GOSUB  NXTELM
2538 145E1 7DF3   FXSA10   GOSUB  FXSTSP         COMPUTE SPACE NEEDED IN FILE
2539 145E5 42D             GOC    FIX350         NOT ENOUGH ROOM
2540 145E8 847             ST=0   Check
2541 145EB 7DB2            GOSUB  NXTELM
2542 145EF 131             D1=A                 D1 @ PAST THE STRING
2543 145F2 D0              A=0    A
2544 145F4 15A3            A=DAT0 4              A= STRING LENGTH IN BYTES
2545 145F8 6ECF            GOTO   FIX360
2546                *
2547 145FC 8C00   =D1mstk  GOLONG =D1MSTK
           00
2548                *
2549                *
```

```
2550            ********************************************************************
2551            ********************************************************************
2552            **
2553            ** Name:    FNDFBF  -  Find File I/O Buffer
2554            **
2555            ** Category:   FILUTL
2556            **
2557            ** Purpose:  When open an external file to FIB, an I/O buffer is
2558            **           allocated to contain one sector of the file. The
2559            **           I/O buffer number is stored in the FIB entry of the
2560            **           file.
2561            **           This routine will find that I/O buffer by given the
2562            **           FIB entry address of the file.
2563            **
2564            ** Entry:  ST#TD1 contains the FIB entry address of the file
2565            **
2566            ** Exit:  B(A) = file I/O buffer start address
2567            **        S10 = 1 if external file
2568            **            = 0 if internal file
2569            **        Error exit to error routine if file I/O buffer not
2570            **          found.
2571            **
2572            ** Calls:  I/OFND
2573            **
2574            ** Uses:  A, B(A), C, D0, S10
2575            **
2576            ** Stk lvls:  1
2577            ********************************************************************
2578            ********************************************************************
2579            *
2580 14602 8EDB =FNDFBF GOSUBL D0=FIB
          4F
2581 14608 161          D0=D0+ (oFBF#b)
2582 1460B 1563         C=DAT0 X
2583 1460F 93A          ?C=0   X              IS THIS AN INTERNAL FILE ?
2584 14612 00           RTNYES                IF SO, RETURN
2585 14614 85A          ST=1   I/Obuf
2586 14617 133          AD1EX                 SAVE D1 IN B(A)
2587 1461A D8           B=A    A
2588 1461C 8E00         GOSUBL =I/OFND
          00
2589 14622 4A0          GOC    FDFBF1
2590 14625 07           C=RSTK                BUFFER NOT FOUND
2591 14627 8C00         GOLONG =F#NFND
          00
2592            *
2593 1462D D4    FDFBF1 A=B    A              RESTORE D1 FROM B(A)
2594 1462F 133          AD1EX
2595 14632 D8           B=A    A              B(A) = BUFFER START ADDRESS
2596 14634 03           RTNCC
2597            *
2598            *
2599            ********************************************************************
2600            ********************************************************************
2601            **
```

```
2602                ** Name:(S) GTPTRS  -  Get File Pointers from FIB
2603                ** Name:(S) GTPTRX  -  Get File Pointers from FIB
2604                **
2605                ** Category:   FILUTL
2606                **
2607                ** Purpose:  Get all the file & FIB pointers into CPU registers
2608                **
2609                ** Entry:    STMTD1(4-0) = Entry address in FIB
2610                **   GTPTRX: Should clear S9 & S10 on entry
2611                **
2612                ** Exit:  D(S) = Copy code of the file
2613                **        D(A) = # of bytes to end of file
2614                **        B(S) = Device type
2615                **        B(A) = # of bytes left in current record
2616                **        R0(A)= Current position (absolute address)
2617                **        R0(15:14) = Relative position in buffer if external
2618                **        R1   = Record length in bytes
2619                **        S9   = 0 if serial acces
2620                **             = 1 if random access
2621                **        S10  = 0 if mainframe RAM/ROM file
2622                **             = 1 if is an external file
2623                **        S11  = 1 if Independent RAM
2624                **             = 0 if not IRAM
2625                **
2626                **   GTPTRX:
2627                **        The difference between the two entry points is that in
2628                **        order to determine whether it is a serial or random
2629                **        access.
2630                **        The GTPTRS entry will go back to the beginning of the
2631                **        statement to check if the record number is specified.
2632                **        But the GTPTRX entry will not do so, therefore the S9
2633                **        will not be changed by the GTPTRX entry.
2634                **
2635                ** Calls:  I/OFND
2636                **
2637                ** Stk lvls:  2
2638                **
2639                ** Used A,B,C,D,D0, S9-11
2640                ********************************************************************
2641                ********************************************************************
2642                *
2643 14636 137   =GTPTRS CD1EX                  SAVE D1 IN D0
2644 14639 136         CD0EX
2645 1463C 1F97        D1=(5) =PCADDR
           6F2
2646 14643 147         C=DAT1 A
2647 14646 135         D1=C   A
2648 14649 175         D1=D1+ 6
2649 1464C 8E00        GOSUBL =EXPSKP
           00
2650 14652 849         ST=0   Random
2651 14655 84A         ST=0   I/Obuf
2652 14658 84B         ST=0   Iram
2653 1465B 14B         A=DAT1 B
2654 1465E 3100        LC(2)  =tSEMIC
```

```
2655 14662 962          ?A=C   B
2656 14665 50           GOYES  GTPTRO
2657 14667 859          ST=1   Random
2658             *
2659 1466A 136   GTPTRO CDOEX                  RESTORE D1 FROM D0
2660 1466D 137          CD1EX
2661 14670 7E8F =GTPTRX GOSUB  FNDFBF         FIND FILE BUFFER IF EXTERNAL FILE
2662 14674 167          D0=D0+ (oCOPYb)-(oFBF#b)
2663 14677 1564         C=DAT0 S
2664 1467B AC7          D=C    S              D(S)= COPY CODE
2665 1467E 161          D0=D0+ (oDEVCb)-(oCOPYb)
2666 14681 1564         C=DAT0 S
2667 14685 AC5          B=C    S              B(S)= DEVICE TYPE
2668 14688 A4E          C=C-1  S              TEST IF IS IRAM
2669 1468B A4E          C=C-1  S
2670 1468E 550          GONC   NOTIRM
2671 14691 85B          ST=1   Iram
2672 14694 168   NOTIRM D0=D0+ (oDBEGb)-(oDEVCb)
2673 14697 146          C=DAT0 A
2674 1469A D7           D=C    A              D(A)= DATA START ADDR (ABS)
2675 1469C 16E          D0=D0+ (oRECLb)-(oDBEGb)
2676 1469F D2           C=0    A
2677 146A1 15E3         C=DAT0 4
2678 146A5 109          R1=C                  R1= RECORD LENGTH
2679 146A8 163          D0=D0+ (oCPOSb)-(oRECLb)
2680 146AB 86A          ?ST=0  I/Obuf         EXTERNAL FILE ?
2681 146AE D2           GOYES  GTPTR2         IF NOT, GOTO GTPTR2
2682 146B0 AF0          A=0    W
2683 146B3 1523         A=DAT0 X
2684 146B7 32EF         LC(3)  510            C(X) = OCT. 776
          1
2685 146BC 0E36         A=A&C  X              A(X) = REL.CUR.POS. IN BUFFER
2686 146C0 D6           C=A    A              C(A) = REL.NIB.POS IN BUFFER
2687 146C2 81C          ASRB                  A(B) = REL. BYTE POS. IN BUFFER
2688 146C5 814          ASRC
2689 146C8 814          ASRC                  A(15:14) = REL. BYTE  POS. IN BUFR
2690 146CB DA           A=C    A
2691 146CD C0           A=A+B  A              A(A)= CURRT POS IN BUFFER (ABS)
2692 146CF 100          R0=A                  R0(15:14)=CURT POS IN BUFR (REL)
2693 146D2 142          A=DAT0 A              A(A) = CURR.POS IN NIBS
2694 146D5 D8           B=A    A              B(A) = CURRENT POSITION(NIBS)
2695 146D7 6D00         GOTO   GTPTR3
2696             *                            R0(A)= CURT POS IN BUFR (ABS)
2697 146DB 146   GTPTR2 C=DAT0 A
2698 146DE D5           B=C    A              B(A)= CURRENT POSITION IN NIBS
2699 146E0 CB           C=C+D  A              C= CURRENT POSITION (ABSOLUTE)
2700 146E2 108          R0=C
2701 146E5 165  GTPTR3  D0=D0+ (oDLENb)-(oCPOSb)
2702 146E8 AF2          C=0    W
2703 146EB 146          C=DAT0 A
2704 146EE E9           C=C-B  A
2705 146F0 411          GOC    GTPTR4
2706 146F3 81E          CSRB
2707 146F6 D7           D=C    A              D= # OF BYTES TO EOF
2708 146F8 165          D0=D0+ (oRLENb)-(oDLENb)
```

```
2709 146FB 146          C=DATO A
2710 146FE D5           B=C    A              B = # OF BYES LEFT IN CUR.RECORD
2711 14700 03           RTNCC
2712              *
2713 14702 8C6E  GTPTR4 GOLONG EOFIL           **Pack here...could save 2 nibs**
          8F
2714              *
2715              ************************************************************
2716              ************************************************************
2717              **
2718              ** Name:(S) STKVCT  -  Process Array Dope Vector
2719              **
2720              ** Category:   EXCUTL
2721              **
2722              ** Purpose: Process an array dope vector on math stack. When
2723              **    printing or reading an array to or from a data file,
2724              **    it is done one element at a time. The array dope vector
2725              **    will remain on the stack until done, so it can be used
2726              **    to keep track of the next element addr and number of
2727              **    elements left to be done.
2728              **    The dope vector on the math stack will contain :
2729              **    Nibs    Meaning
2730              **    -----   ------------------------------------------
2731              **    0       Variable type. A-Int, B-Short, C-Real....
2732              **    1       Dimensions. (1 or 2)
2733              **    2       Option base.
2734              **    3-6     Maximum string length if is string variable
2735              **    7-10    Number of elements left to be done.
2736              **    11-15   Next element address.
2737              **
2738              ** Entry:  D1 @ stack pointer
2739              **         If S8 =1, rewite dope vector
2740              ** EXIT:
2741              **         Following status bit will be set properly :
2742              **         Notnum(S0) - Not simple real
2743              **         Array (S1) - Numeric or String array
2744              **         String(S2) - String or string vector
2745              **         Cmplex(S3) - Complex number or Complex array
2746              **         If is an array element(S1=1):
2747              **            Carry clear => All elements done
2748              **
2749              **            STACK(10-7)= Number of elements left
2750              **            STACK(6-3) = String max. length in bytes
2751              **            STACK(15-11)= Next element address
2752              ** Used:   A,C,D0,D1
2753              ************************************************************
2754              ************************************************************
2755              *
2756 14708 70FE =STKVC+ GOSUB  D1mstk
2757 1470C 1537 =STKVCT A=DAT1 W              READ THE STACK
2758 14710 30A          LCHEX  A
2759 14713 982          ?A<C   P              SIMPLE REAL ON STACK ?
2760 14716 00           RTNYES                IF SO, JUST RETURN
2761 14718 850          ST=1   Notnum
2762 1471B 852          ST=1   String         ASSUME IS STRING
```

```
2763 1471E 30E          LCHEX  E
2764 14721 986          ?A>C   P              IF A(0) = F, IS A STRING
2765 14724 01           GOYES  STKV10
2766 14726 842          ST=0   String
2767 14729 A0E          C=C-1  P              C(0) = D
2768 1472C 982          ?A<C   P
2769 1472F 50           GOYES  STKV10
2770 14731 853          ST=1   Cmplex         IF A(0)=E OR D, IS COMPLEX
2771 14734 AE6  STKV10  C=A    B
2772 14737 BE6          CSR    B
2773 1473A 90A          ?C=0   P
2774 1473D 00           RTNYES                NOT AN ARRAY
2775 1473F 851          ST=1   Array
2776 14742 878          ?ST=1  Fresh
2777 14745 21           GOYES  NEWITM
2778 14747 176  STKV15  D1=D1+ 7
2779 1474A D2           C=0    A
2780 1474C 15F3         C=DAT1 4              READ ELEMENT COUNT
2781 14750 8AE          ?C#0   A              COUNT # 0 ?
2782 14753 00           RTNYES
2783 14755 03           RTNCC                 IF COUNT = 0 ,CLEAR CARRY
2784 14757 1BCD NEWITM  D0=(5) (=SYSFLG)+3
           6F2
2785 1475E A06          C=C+C  P
2786 14761 5F3          GONC   STKV40
2787            * NONEXIST ARRAY
2788 14764 AF2          C=0    W
2789 14767 1564         C=DAT0 S              C(S) = OPTION BASE
2790 1476B 30A          LC(1)  10
2791 1476E A46          C=C+C  S              OPTION BASE = 1 ?
2792 14771 440          GOC    STKV20
2793 14774 E6           C=C+1  A              11 ELEMENTS IF OPTION BASE 0
2794 14776 1BAB STKV20  D0=(5) =F-R1-3
           8F2
2795 1477D 1522         A=DAT0 XS
2796 14781 A2C          A=A-1  XS             A(XS) = SUBSCRIPT COUNT
2797 14784 421          GOC    STKV25
2798 14787 A2C          A=A-1  XS
2799 1478A 4C0          GOC    STKV25
2800 1478D DA           A=C    A              SUBSCRIPT COUNT = 2
2801 1478F 8E00         GOSUBL =A-MULT
           00
2802 14795 D6           C=A    A
2803 14797 176  STKV25  D1=D1+ 7
2804 1479A 15D8         DAT1=C 9              WRITE ELEMENT NUMBER
2805 1479E 521          GONC   STKV50
2806            *
2807 147A1 872  STKV40  ?ST=1  String
2808 147A4 92           GOYES  STKV60
2809 147A6 8F00         GOSBVL =ARYELM        COMPUTE # OF ELEMENTS
           000
2810 147AD 1593 STKV45  DAT1=A 4              WRITE # OF ELEMENTS
2811            *
2812 147B1 173  STKV50  D1=D1+ 4              D1 @ ARRAY ADDRESS
2813 147B4 147          C=DAT1 A              C = ARRAY START ADDRESS
```

```
2814                *
2815 147B7 1B17             D0=(5) =S-R0-0
           8F2
2816 147BE 144              DAT0=C A              WRITE ARRAY ADDRESS TO S-R0-0
2817 147C1 16F              D0=D0+ 16             INITIALIZE THE REG # FOR TRACE
2818 147C4 D2               C=0    A
2819 147C6 CE               C=C-1  A              INITIALIZE TO FFFFF
2820 147C8 144              DAT0=C A
2821 147CB 02               RTNSC
2822                *
2823 147CD 147   STKV60      C=DAT1 A              C(XS) = OPTION BASE
2824 147D0 176              D1=D1+ 7
2825 147D3 15B3             A=DAT1 4
2826 147D7 0B               CSTEX                 S8= OPTION BASE OF THIS VECTOR
2827 147D9 878              ?ST=1  8              **Pack here...could save 3 nibs**
2828 147DC 20               GOYES  STKV70         ****(CSRB, C=C+C B)****
2829 147DE 0B    STKV70     CSTEX
2830 147E0 4CC              GOC    STKV45         IF SO, ELEMENT NUMBER IS RIGHT
2831 147E3 E4               A=A+1  A
2832 147E5 57C              GONC   STKV45         (B.E.T.)
2833                *
2834                ********************************************************************
2835                ********************************************************************
2836                **
2837                ** Name:(S) NXTADR  -  Get Address of Next Array Element
2838                **
2839                ** Category:   EXCUTL
2840                **
2841                ** Purpose:  Get the address of next element of an array
2842                **
2843                ** Entry:
2844                **        MTHSTK pts at the array dope vector(top of stack)
2845                **        S8 = 1 If to get the address of the first element
2846                **           When the dope vector is first time recalled to
2847                **           the math stack, the address field already
2848                **           point to the next element address. Set S8 will
2849                **           it been moved to next element address.
2850                **
2851                ** Exit:  Carry clear:
2852                **          D1 @ Top of stack
2853                **          S-R0-3 = Data type:  0- real, 1-short, 2-integer
2854                **                   E- complex, F- short complex, D-STRING
2855                **          S-R0-0 = next element address
2856                **          If is a string vector:
2857                **           R3 = Max. string length
2858                **           S-R1-1 = Max. string length
2859                **
2860                ** Used A,C,D0,D1, STMTR0, STMTR1, R3 (if string vector)
2861                **
2862                ** Stk lvls: 1
2863                ********************************************************************
2864                ********************************************************************
2865                *
2866 147E8 72A0 =NXTADR GOSUB  NXTECK
2867 147EC 1593             DAT1=A 4              UPDATE ELEMENT COUNT
```

```
2868 147F0 810          ASLC   W          A(0) = DATA TYPE
2869 147F3 1C3          D1=D1- 4
2870 147F6 D2           C=0    A
2871 147F8 862          ?ST=0  String
2872 147FB 54           GOYES  NXTA10
2873 147FD 15F3         C=DAT1 4           C = MAX. STRING LNEGTH
2874 14801 10B          R3=C               SAVE MAX. LENGTH IN R3
2875 14804 1B17         D0=(5) =S-R0-0
           8F2
2876 1480B 142          A=DAT0 A           A = DESTINATION ADDRESS
2877 1480E 16F          D0=D0+ 16
2878 14811 164          D0=D0+ 5           D0 @ S-R1-1
2879 14814 144          DAT0=C A           WRITE MAX. LENGTH TO S-R1-1
2880 14817 E6           C=C+1  A           ADD HEADER LENGTH(2 BYTES)
2881 14819 E6           C=C+1  A
2882 1481B C6           C=C+C  A
2883 1481D CA           A=A+C  A           A= NEXT ELEMENT ADDRESS
2884 1481F 18F          D0=D0- 16          D0 @ S-R0-1
2885 14822 AF2          C=0    W
2886 14825 2E           P=     14
2887 14827 80FA         CPEX   10
2888 1482B A7E          C=C-1  W
2889 1482E D2           C=0    A           C(10,0) = DFFFFF00000
2890 14830 15CA         DAT0=C 11
2891 14834 184          D0=D0- 5           D0 @ S-R0-0
2892 14837 5E3          GONC   NXTA20      (B.E.T.)
2893 1483A FEF0 =DATCOD NIBHEX FEF012
           12
2894 14840 30F  NXTA10  LCHEX  F
2895 14843 B02          C=C-A  P
2896 14846 DA           A=C    A           A(A)=0-4 FOR F-A
2897 14848 34A3         LC(5)  =DATCOD
           841
2898 1484F C2           C=A+C  A
2899 14851 134          D0=C
2900 14854 1524         A=DAT0 S           READ THE DATA CODE
2901 14858 D6           C=A    A
2902 1485A 8F00         GOSBVL =DATLEN
           000
2903 14861 20           P=     0
2904 14863 1B08         D0=(5) =S-R0-3
           8F2
2905 1486A 1504         DAT0=A S
2906 1486E 18E          D0=D0- 15          D0 @ S-R0-0
2907 14871 142          A=DAT0 A           A = DESTINATION ADDRESS
2908 14874 CA           A=A+C  A           A = NEXT ELEMENT ADDRESS
2909            *
2910 14876 1C2  NXTA20  D1=D1- 3           D1 @ TOP OF STACK
2911 14879 878          ?ST=1  Fresh
2912 1487C 50           GOYES  NXTA30
2913 1487E 140          DAT0=A A           WRITE NEXT ELEMENT ADDRESS
2914 14881 16F  NXTA30  D0=D0+ 16           D0 @ S-R1-0
2915 14884 146          C=DAT0 A           INCREASE REG # FOR TRACE
2916 14887 E6           C=C+1  A
2917 14889 144          DAT0=C A
```

```
2918 1488C 03           RTNCC
2919            *
2920            ****************************************************************
2921            ****************************************************************
2922            ** Name:(S) NXTELM  -  Get Next Array Element
2923            **
2924            ** Category: EXCUTL
2925            **
2926            ** Purpose:  Get next array element
2927            **           While printing or reading an array, the array
2928            **           vector on the stack is used to keep track of next
2929            **           element address and # of elements left. This
2930            **           routine will get the next element and update the
2931            **           vector information.
2932            **
2933            ** Entry:
2934            **    The dope vector on the math stack will contain :
2935            **    Nibs    Meaning
2936            **    -----   ------------------------------------------------
2937            **    0       Variable type. A-Int, B-Short, C-Real, D-S.Complex
2938            **                           E- Complex, F- String
2939            **    1       Dimensions. (1 or 2)
2940            **    2       Option base.
2941            **    3-6     Maximum string length if this is string variable
2942            **    7-10    Number of elements left to be done.
2943            **    11-15   Next element address.
2944            **
2945            **
2946            ** Exit:  Carry set => All done, there is no next element
2947            **        Carry clear => Not done yet, there are more elements.
2948            **           S5 = 1 if no room on math stack to recall the value
2949            **                  of next element.
2950            **           If numeric array:
2951            **             B = Next element
2952            **             The element count and next element address will
2953            **               be updated in the array dope vector on math
2954            **               stack.
2955            **             If still room on stack, the element will be
2956            **               written to the stack on top of the array dope
2957            **               vector and the MTHSTK will be updated
2958            **           If is a complex array:
2959            **             D = Imaginary part
2960            **             B = Real part
2961            **             The two number will be written to stack too
2962            **           If string:  DO @ string start
2963            **                       A= Address past the string element
2964            **                       C= String length in nibs + 4
2965            **
2966            ** Note:
2967            **        The data type, such as real, string or complex, should
2968            **        still be indicated by S2 and S3 :
2969            **        S2 = 1 - String
2970            **        S3 = 1 - Complex
2971            **
2972            ** Used:  A,B,C
```

```
2973                **
2974                ** Stk lvls: 1
2975                ************************************************************************
2976                ************************************************************************
2977                *
2978 1488E 1F99  NXTECK D1=(5) =MTHSTK
           5F2
2979 14895 143          A=DAT1 A
2980 14898 131          D1=A
2981 1489B 1534         A=DAT1 S
2982 1489F 176          D1=D1+ 7
2983 148A2 D0           A=0    A
2984 148A4 15B3         A=DAT1 4              A= # OF ELEMENTS LEFT
2985 148A8 CC           A=A-1  A              ALL DONE WITH THE ARRAY ?
2986 148AA 01           RTN
2987 148AC 7EDF =NXTELM GOSUB  NXTECK         **Pack here...could save 3 nibs**
2988 148B0 867          ?ST=0  Check
2989 148B3 81           GOYES  NXTE05
2990                * If just for checking, return with A(A)= next element length
2991                * Check can only be used by string
2992 148B5 173          D1=D1+ 4
2993 148B8 143          A=DAT1 A
2994 148BB 8A8          ?A=0   A
2995 148BE B0           GOYES  NXTE02
2996 148C0 131          D1=A                  D1 @ NEXT STRING ELEMENT ADDRESS
2997 148C3 D0           A=0    A
2998 148C5 15B3         A=DAT1 4              A= NEXT ELEMENT LENGTH
2999 148C9 03    NXTE02 RTNCC
3000 148CB 1593  NXTE05 DAT1=A 4              UPDATE ELEMENT COUNT
3001 148CF 810          ASLC   W              A(0) = DATA TYPE
3002 148D2 D2           C=0    A
3003 148D4 862          ?ST=0  String
3004 148D7 34           GOYES  NXTE20
3005 148D9 1C3          D1=D1- 4
3006 148DC 15F3         C=DAT1 4              C= STRING MAX. LENGTH
3007 148E0 177          D1=D1+ 8
3008 148E3 143          A=DAT1 A              A=NEXT ELEMENT ADDRESS
3009 148E6 8AC          ?A#0   A              ARRAY EXIST ?
3010 148E9 01           GOYES  NXTE12         IF SO, GOTO NXTE12
3011 148EB 137          CD1EX                 FAKE THE STRING ADDRESS
3012 148EE 135          D1=C
3013 148F1 134          D0=C                  MAKE STRING POINTS TO EMPTY VECTOR
3014 148F4 D2           C=0    A
3015 148F6 5E1          GONC   NXTE15         (B.E.T.)
3016 148F9 130   NXTE12 D0=A
3017 148FC E6           C=C+1  A             ADD TWO BYTES OF STRING LENGTH
3018 148FE E6           C=C+1  A
3019 14900 C6           C=C+C  A             BECOME NIBBLES
3020 14902 C2           C=A+C  A              POINTS TO NEXT ELEMENT
3021 14904 145          DAT1=C A
3022 14907 D2           C=0    A
3023 14909 15E3         C=DAT0 4              C= STRING LENGTH
3024 1490D E6           C=C+1  A
3025 1490F E6           C=C+1  A
3026 14911 C6           C=C+C  A
```

```
3027 14913 CA          A=A+C  A            A @ PAST STRING
3028 14915 1CA  NXTE15 D1=D1- 11
3029 14918 03          RTNCC
3030 1491A 30F  NXTE20 LCHEX  F
3031 1491D B02         C=C-A  P            C(0)= 0-5 FOR F-A
3032 14920 8F00        GOSBVL =DATLEN
           000
3033 14927 20          P=     0
3034 14929 173         D1=D1+ 4
3035 1492C 143         A=DAT1 A            A= NEXT ELEMENT ADDRESS
3036 1492F 8AC         ?A#0   A            ARRAY EXIST ?
3037 14932 B0          GOYES  NXTE25       IF SO, GOTO NXTE25
3038 14934 1CA         D1=D1- 11
3039 14937 AF2         C=0    W
3040 1493A 564         GONC   NXTE40
3041 1493D 130  NXTE25 DO=A
3042 14940 CA          A=A+C  A
3043 14942 141         DAT1=A A            UPDATE NEXT ELEMENT ADDRESS
3044 14945 1CA         D1=D1- 11
3045 14948 DA   =RCLNUM A=C    A
3046 1494A 3101        LCHEX  10
3047 1494E 9EE         ?A>=C  B            REAL OR COMPLEX ?
3048 14951 85          GOYES  NXTREL
3049             * INTEGER OR SHORT
3050 14953 29   NXTE30 P=     9
3051 14955 189         DO=DO- 10
3052 14958 1567        C=DATO W
3053 1495C A92         C=0    WP
3054 1495F 20          P=     0
3055 14961 A04         A=A+A  P
3056 14964 4A3         GOC    NXTSHT
3057 14967 AFA         A=C    W            SAVE D IN A TEMP.
3058 1496A AFF         CDEX   W
3059 1496D AFE         ACEX   W
3060 14970 8E00        GOSUBL =READIN
           00
3061 14976 AFE         ACEX   W            RESTORE D FROM A
3062 14979 AF7         D=C    W
3063 1497C AFE         ACEX   W
3064 1497F 20          P=     0
3065 14981 AF5  NXTE40 B=C    W
3066 14984 8E92        GOSUBL STK16?
           4F
3067 1498A 5F0         GONC   NXTNOR
3068 1498D AF9         C=B    W
3069 14990 1557        DAT1=C W
3070 14994 8CF5 upmths GOLONG UPMTHS
           4F
3071 1499A 855  NXTNOR ST=1   Error
3072 1499D 03          RTNCC
3073 1499F 16F  NXTSHT DO=DO+ 16
3074 149A2 1563        C=DATO X
3075 149A6 5AD         GONC   NXTE40       (B.E.T.)
3076 149A9 9E6  NXTREL ?A>C   B            COMPLEX NUMBER ?
3077 149AC 90          GOYES  NXTE60
```

```
3078 149AE 1567  NXTE50 C=DAT0 W
3079 149B2 5EC          GONC   NXTE40          (B.E.T.)
3080 149B5 3121  NXTE60 LCHEX  12
3081 149B9 853          ST=1   Cmplex          Set complex flag again
3082 149BC 9E6          ?A>C   B               COMPLEX REAL ?
3083 149BF 41           GOYES  NXTE70
3084 149C1 81C          ASRB
3085 149C4 7B8F         GOSUB  NXTE30
3086 149C8 162          DO=DO+ 3
3087 149CB DO           A=0    A
3088 149CD CC           A=A-1  A               A(0)= F
3089 149CF 638F         GOTO   NXTE30          GET THE REAL PART(B.E.T.)
3090 149D3 1567  NXTE70 C=DAT0 W
3091 149D7 76AF         GOSUB  NXTE40
3092 149DB 16F          DO=DO+ 16
3093 149DE 6FCF         GOTO   NXTE50
3094            *
3095            *
3096            ************************************************************************
3097            ************************************************************************
3098            **
3099            ** Name: FXSTSP  -  Get Space to Store String in a DATA file
3100            **
3101            ** Purpose:  The record length in the DATA file is fixed, so
3102            **           if the string length is longer than the record
3103            **           length, the string be stored in multiple records.
3104            **           This routine will compute the total number of
3105            **           bytes to store the entire string in the file.
3106            **           Serial access is assumed by this routine.
3107            **
3108            ** Entry:  B= # of bytes left in current record
3109            **         D= # of bytes left in current file
3110            **         A= String length in bytes
3111            **         If the file need to be expanded, the S4 will
3112            **          determine where the routine should return to
3113            **          after expanding the file:
3114            **         S4 = 1 - Return to the calling routine.
3115            **         S4 = 0 - Exit to the main loop of PRINT# code
3116            **                  (PRT220).
3117            **
3118            ** Exit:  Carry set => Not enough room to expand current file
3119            **        Carry clear => Current file has enough room
3120            **        C= Total # of bytes needed
3121            **        D= # of bytes left in the file afterward
3122            **
3123            **        If the current file doesn't have enough room to store
3124            **        the string, but the file is in RAM, this routine will
3125            **        fall into the GETSPC routine in an attempt to expand
3126            **        the file.
3127            **
3128            ** Uses:  A,C,D,R2,D1
3129            **
3130            ** Stk lvls: 3
3131            ************************************************************************
3132            ************************************************************************
```

```
3133                *
3134 149E2 102   FXSTSP R2=A                SAVE A IN R2 TEMP.
3135 149E5 D2           C=0    A
3136 149E7 303          LC(1)  3
3137 149EA 111          A=R1                A= RECORD LENGTH
3138 149ED 8B6          ?A>C   A            RECORD LENGTH > 3
3139 149F0 60           GOYES  FXSP10
3140 149F2 6E2B  FXSP05 GOTO   RECERR       SAY "RECORD ERROR"
3141 149F6 D4    FXSP10 A=B    A
3142 149F8 122          AR2EX               SAVE B IN R2
3143 149FB 8BD          ?B<=C  A            CURRENT RECORD TRAIL < 4 ?
3144 149FE 21           GOYES  FXSP20       IF SO, SKIP OVER TO NEXT RECORD
3145 14A00 C2           C=A+C  A            C= STRING LENGTH +3
3146 14A02 8B9          ?B>=C  A            WHOLE STRING FIT IN CURRENT RECORD?
3147 14A05 D2           GOYES  FXSP70       IF SO, BYTE COUNT= STRING LEN +3
3148 14A07 E9           C=C-B  A
3149 14A09 DA           A=C    A            A= REMAIN STRING LEN FOR NEXT RECORD
3150 14A0B D2           C=0    A
3151 14A0D 303          LC(1)  3
3152 14A10       FXSP20
3153               ******************************
3154               * Bug fix by NZ on 8/18/83...
3155               *
3156               * This fixes the problem of random PRINT# to a file with a
3157               * string which would require the file to be expanded (bug is
3158               * that the fileIS expanded before the check for record overflow,
3159               * leaving some uninitialized records at the end of the file)
3160               *
3161               * One pack was made to balance 2 of these 5 nibbles...see about
3162               * 40 lines above this point (end of routine moved to TI&FX3 for
3163               * rest).
3164               *
3165 14A10 879           ?ST=1  Random        Is this a random print?
3166 14A13 FD          GOYES  FXSP05         Yes, error (exceed current rec)
3167               *
3168               * End of bug fix by NZ
3169               ******************************
3170 14A15 CA          A=A+C  A              A= REMAIN STRING LENGTH + 3
3171 14A17 119   FXSP30 C=R1
3172 14A1A DD          BCEX   A              C= BYTES COUNT
3173 14A1C E0    FXSP40 A=A-B  A
3174 14A1E 4D0         GOC    FXSP50
3175 14A21 E4          A=A+1  A
3176 14A23 E4          A=A+1  A
3177 14A25 E4          A=A+1  A
3178 14A27 C9          C=C+B  A
3179 14A29 52F         GONC   FXSP40         (B.E.T.)
3180 14A2C C0    FXSP50 A=A+B  A
3181 14A2E CA          A=A+C  A
3182 14A30 D6          C=A    A
3183 14A32 87A   FXSP70 ?ST=1  I/Obuf
3184 14A35 40          GOYES  FXSP80
3185 14A37 E6          C=C+1  A              Internal file need one more byte
3186 14A39 8B7   FXSP80 ?D<C   A
3187 14A3C 33          GOYES  GETS10
```

```
3188 14A3E D6          C=A    A
3189 14A40 6828        GOTO   FXSP90          Restore B(A) from R2
3190             *
3191             ****************************************************************
3192             ****************************************************************
3193             ** Name:    STRSPC  -  Get Space to Store String in TEXT file.
3194             **
3195             ** Purpose:  Check if there is enough room left in the TEXT
3196             **           file for appending a new string. If not, try to
3197             **           expand the file.
3198             **
3199             ** Entry:  D1 @ top of the stack pointing at the string
3200             **         D(A) = # of bytes left in the current file
3201             **         STMTD1 = FIB entry address of the file
3202             **         If the file need to be expanded, the S4 will
3203             **          determine where the routine should return to
3204             **          after expanding the file:
3205             **         S4 = 1 - Return to the calling routine.
3206             **         S4 = 0 - Exit to the main loop of PRINT# code
3207             **                  (PRT220).
3208             ** Exit:
3209             **         Carry set => not enough room to expand
3210             **         Carry clear => enough room or been expanded
3211             **
3212             **        If the current file doesn't have enough room to store
3213             **        the string, but the file is in RAM, this routine will
3214             **        fall into the GETSPC routine in an attempt to expand
3215             **        the file.
3216             **
3217             ** Used:  A, C - if no need to expand the file
3218             **        A,B,C,D,D0,D1,R0 - if need to expand the file
3219             ****************************************************************
3220             ****************************************************************
3221             *
3222 14A44 171  =STRSPC D1=D1+ 2
3223 14A47 AF0          A=0    W
3224 14A4A 143          A=DAT1 A              READ THE STRING LENGTH
3225 14A4D 81C          ASRB                 STRING LENGTH IN BYTES
3226             ******************************
3227             * Pack here...could save 1 nibble
3228             *
3229             *STSPC- C=0    A
3230             *       C=C+1  A
3231             *
3232 14A50 301   STSPC- LC(1)  1
3233 14A53 0E02        C=A&C  P
3234 14A57 90A         ?C=0   P              STRING LENGTH EVEN
3235 14A5A 70          GOYES  STRS10
3236 14A5C 856         ST=1    Odd
3237 14A5F E4          A=A+1  A              STRING LENGTH ODD
3238 14A61 D2    STRS10 C=0    A             STRING LENGTH + HEADER(2)
3239             *
3240             *STRS10                      STRING LENGTH + HEADER(2)
3241             *
3242             * End of possible pack
```

```
3243            ******************************
3244 14A63 304          LC(1)  4                +EOF MARK(2)
3245 14A66 C2           C=A+C  A              C=TOTAL SPACE IN FILE
3246            *
3247            ************************************************************************
3248            ************************************************************************
3249            **
3250            ** Name:    GETSPC  -  Make Space in File for Write
3251            **
3252            ** Category:   FILUTL
3253            **
3254            ** Purpose:  Find out if there is enough room left in the
3255            **           current file for a write. If there is not enough
3256            **           room left, try to expand the file.
3257            **
3258            ** Entry:  D(A) = # of bytes left in current file
3259            **         C(A) = # of bytes required
3260            **         STMTD1 = Entry address in FIB
3261            **         If the file need to be expanded, S4 will be used to
3262            **          determine where the routine should return to after
3263            **          the file has been expanded:
3264            **          S4 = 1 - Return to calling routine.
3265            **             = 0 - Exit to the main loop of PRINT# code
3266            **                   (PRT220).
3267            **
3268            ** Exit:   Carry clear => There is enough room left in the file
3269            **         Carry set => Not enough room to expand the file
3270            **         If file has been expanded and S4=0, exit to PRT220
3271            **
3272            **     If the current file doesn't have enough room to store
3273            **     the string, but the file is in RAM, this routine will
3274            **     try to expand the file.
3275            **     If the expansion is successful, the S4 will be used
3276            **     to determine where this routine should return to.
3277            **     If the S4 = 0, this routine will exit to PRT220
3278            **     directly.
3279            **     The reason for not returning to calling routine is that
3280            **     after expanding file, most of the information kept in the
3281            **     CPU registers are lost, such as data type, file pointer,
3282            **     record size, etc. So directly exit to PRT220 will cause
3283            **     the whole process of printing next data item to the file
3284            **     starting over again.
3285            **
3286            ** Uses:  Nothing, if still room left
3287            **        A,B,C,D,D0,D1,R0,R1,R3,S2 If need to expand the file
3288            **
3289            ** Stk lvls : 3
3290            ************************************************************************
3291            ************************************************************************
3292            *
3293 14A68 8B7  =GETSPC ?D<C   A
3294 14A6B 40           GOYES  GETS10
3295 14A6D 03           RTNCC
3296 14A6F 87A   GETS10 ?ST=1  I/Obuf
3297 14A72 00           RTNYES                NO EXPANSION OF EXTERNAL FILE
```

```
3298 14A74 EB          C=C-D  A
3299 14A76 C6          C=C+C  A              C= EXTRA NIBBLES REQUIRED
3300 14A78 AF1         B=0    W
3301 14A7B D5          B=C    A
3302 14A7D 86B         ?ST=0  Iram           MAINFRAME RAM ?
3303 14A80 A6          GOYES  GETS12         IF SO, GOTO GETS20
3304 14A82 1F18        D1=(5) =S-R1-0        SAVE # OF NIBS REQUIRED IN S-R1-0
           8F2
3305 14A89 145         DAT1=C A
3306 14A8C 1F69        D1=(5) =STMTD1
           8F2
3307 14A93 143         A=DAT1 A              A= FIB ENTRY ADDRESS
3308 14A96 D2          C=0    A
3309 14A98 31C1        LC(2)  (oDBEGb)+7     OFFSET TO PORT #
3310 14A9C CA          A=A+C  A
3311 14A9E 131         D1=A
3312 14AA1 147         C=DAT1 A
3313 14AA4 AE7         D=C    B              D(B) = PORT #
3314 14AA7 8F00        GOSBVL =ROMF-         FIND THE PORT HAS THE IRAM
           000
3315 14AAE 400         RTNC                  NOT FOUND, SOMETHING WRONG !
3316 14AB1 137         CD1EX                 C= FILE CHAIN START ADDRESS
3317 14AB4 8F00        GOSBVL =EOFLC+        LOOK FOR END OF FILE CHAIN
           000
3318 14ABB E6          C=C+1  A
3319 14ABD E6          C=C+1  A              PAST THE 00 BYTE
3320 14ABF 1F68        D1=(5) =S-R1-1
           8F2
3321 14AC6 145         DAT1=C A              SAVE CHAIN END IN S-R1-1
3322 14AC9 8E00        GOSUBL =LSTADR        COMPUTE LAST ADDR OF THE PORT
           00
3323 14ACF 143         A=DAT1 A              A= CHAIN END ADDR
3324 14AD2 DE          ACEX   A              STORE CHAIN END TO D(A)
3325 14AD4 D7          D=C    A
3326 14AD6 DE          ACEX   A
3327 14AD8 E2          C=C-A  A              C= # OF NIBS AVAILABLE
3328 14ADA 1C4         D1=D1- 5
3329 14ADD 143         A=DAT1 A              A= # OF NIBS NEEDED
3330 14AE0 D8          B=A    A
3331 14AE2 8B6         ?C<A   A              ENOUGH ROOM TO EXPAND ?
3332 14AE5 00          RTNYES                IF NOT, RETURN WITH CARRY SET
3333            * D(A) = END OF FILE CHAIN ADDRESS( EQUIV. AVMEMS)
3334            * B(A) = # OF NIBS NEEDED
3335            * S-R1-1  CONTAINS CHAIN END ADDR
3336            *
3337 14AE7 591          GONC   GETS15         (B.E.T.)
3338            *
3339 14AEA 8E00  GETS12 GOSUBL =chkspc
           00
3340 14AF0 400         RTNC                 NO ROOM TO EXPAND, RETURN CARRY SET
3341 14AF3 D6          C=A    A
3342 14AF5 D7          D=C    A              D= AVMEMS
3343 14AF7 1F68        D1=(5) =S-R1-1        SAVE CUT OFF POINTER IN S-R1-1
           8F2
3344 14AFE 145         DAT1=C A
```

```
3345 14B01 1F69  GETS15 D1=(5) =STMTD1
           8F2
3346 14B08 143          A=DAT1 A
3347 14B0B 131          D1=A                  D1 @ FIB ENTRY ADDRESS
3348 14B0E 179          D1=D1+ (oCOPYb)
3349 14B11 1534         A=DAT1 S              A(S) = COPY CODE
3350 14B15 172          D1=D1+ (oFBEGb)-(oCOPYb)
3351 14B18 143          A=DAT1 A
3352 14B1B 130          D0=A                  D0 @ FILE HEADER START
3353 14B1E 16F          D0=D0+ (oFTYPh)
3354 14B21 16F          D0=D0+ (oFLENh)-(oFTYPh)
3355 14B24 142          A=DAT0 A              A= CURRENT FILE CHAIN LENGTH
3356 14B27 D6           C=A    A              C=          //
3357 14B29 C0           A=A+B  A              A= NEW FILE CHAIN LENGTH
3358 14B2B 140          DAT0=A A              UPDATE FILE CHAIN LENGTH
3359 14B2E 177          D1=D1+ (oDBEGb)-(oFBEGb)
3360 14B31 17A          D1=D1+ (oREC#b)-(oDBEGb)
3361 14B34 17D          D1=D1+ (oDLENb)-(oREC#b)
3362 14B37 143          A=DAT1 A              UPDATE FILE DATA LENGTH IN FIB
3363 14B3A C0           A=A+B  A
3364 14B3C 141          DAT1=A A
3365 14B3F 132          ADOEX
3366 14B42 101          R1=A                  SAVE COPY CODE & FILE CHAIN ADDR.
3367 14B45 CA           A=A+C  A              A @ CURRENT FILE END
3368 14B47 DB           C=D    A              C= AVMEMS
3369 14B49 E2           C=C-A  A              C= SIZE OF SOURCE TO MOVE
3370 14B4B DF           CDEX   A              D=          //
3371 14B4D 134          D0=C                  D0 @ END SOURCE ADDR (AVMEMS)
3372 14B50 DA           A=C    A
3373 14B52 C0           A=A+B  A
3374 14B54 131          D1=A                  D1 @ END DEST ADDR(AVMEMS+EXPAND
3375 14B57 DF           CDEX   A                                 SIZE)
3376 14B59 8E00         GOSUBL =MOVED3
           00
3377 14B5F 132          ADOEX                 A= OLD END OF FILE ADDRESS
3378 14B62 100          R0=A
3379 14B65 1F68         D1=(5) =S-R1-1
           8F2
3380 14B6C 8F00         GOSBVL =RFAD+I        ADJUST POINTERS
           000
3381 14B73 1B19         D0=(5) =STMTD0        ADJUST ENTRY ADDRESS IN FIB
           8F2
3382 14B7A 8F00         GOSBVL =RFUPD+
           000
3383 14B81 164          D0=D0+ 5
3384 14B84 8F00         GOSBVL =RFUPD+
           000
3385 14B8B 111          A=R1
3386 14B8E A4C          A=A-1  S
3387 14B91 A4C          A=A-1  S
3388 14B94 505          GONC   GETS50
3389              * RECOMPUTE # OF RECORDS IN THIS FILE FOR THE
3390              * FIXED LENGTH DATA FILE
3391 14B97 130          D0=A                  D0 @ FILE CHAIN OF FILE HEADER
3392 14B9A AF2          C=0    W
```

```
3393 14B9D 30D          LC(1)  13
3394 14BA0 AF0          A=0    W
3395 14BA3 142          A=DATO A              A= FILE CHAIN LENGTH IN NIBS
3396 14BA6 EA           A=A-C  A
3397 14BA8 168          DO=DO+ 9              DO @ RECORD LENGTH
3398 14BAB 15E3         C=DATO 4
3399 14BAF C6           C=C+C  A              C= RECORD LENGTH IN NIBS
3400 14BB1 8E00         GOSUBL =IDIV
           00
3401 14BB7 20           P=     0
3402 14BB9 8A9          ?B=0   A
3403 14BBC 40           GOYES  GETS40
3404 14BBE E4           A=A+1  A
3405 14BC0 D8    GETS40 B=A    A
3406 14BC2 183          DO=DO- 4
3407 14BC5 1583         DATO=A 4
3408             * UPDATE # OF RECORDS IN FIB
3409 14BC9 1B69         DO=(5) =STMTD1
           8F2
3410 14BD0 142          A=DATO A
3411 14BD3 3402         LC(5)  (oREC#b)       **Pack here...could save 1 nib****
           000
3412 14BDA CA           A=A+C  A
3413 14BDC 130          DO=A
3414 14BDF D9           C=B    A
3415 14BE1 15C3         DATO=C 4
3416             *
3417 14BE5 731A  GETS50 GOSUB  D1mstk
3418 14BE9 1537         A=DAT1 W
3419 14BED 864          ?ST=0  Return
3420 14BF0 40           GOYES  GETS60
3421 14BF2 03           RTNCC
3422 14BF4 07    GETS60 C=RSTK                POP OFF THE RETUR ADDESSS
3423             ********************************
3424             * Pack here...could save 2 nibbles
3425             *
3426             *       GOTO   PRT220
3427             *
3428 14BF6 8C31         GOLONG PRT220
           8F
3429             *
3430             * End of possible pack
3431             ************************************
3432             *
3433             * N-STR:  CONVERT HP41C REGISTER INTO HP-71 STRING
3434             *
3435             * ENTRY:  D1 PAST THE NUMBER ON STACK
3436             *
3437             * EXIT:  STRING DATA ON STACK
3438             *        D1 @ STRING HEADER
3439             *
3440             * USES:  C(A), A, DO, D1
3441             *
3442 14BFC 137    N-STR  CD1EX
3443 14BFF 134          DO=C
```

```
3444 14C02 135          D1=C
3445 14C05 181          DO=DO- 2              SKIP OVER SIGN AND MO
3446 14C08 2B           P=     11             4 BYTES IN MANTISSA
3447 14C0A 0C    N-ST10 P=P+1
3448 14C0C 490          GOC    N-ST20
3449 14C0F 7A50         GOSUB  R&SKP+
3450 14C13 46F          GOC    N-ST10         (B.E.T.)
3451              *
3452 14C16 180   N-ST20 DO=DO- 1
3453 14C19 1560         C=DATO P
3454 14C1D F2           CSL    A
3455 14C1F 182          DO=DO- 3
3456 14C22 1560         C=DATO P
3457 14C26 7940         GOSUB  R&SKP-
3458 14C2A 7F30  N-ST30 GOSUB  R&SKP+
3459              *
3460              *
3461              ************************************************************
3462              ************************************************************
3463              **
3464              ** Name:(S) STRHED  -  Generate String Head on Stack
3465              **
3466              ** Category:   EXCUTL
3467              **
3468              ** Purpose: Generates string header on stack
3469              **
3470              ** Entry:
3471              **      The string data is sitting on top of MTHSTK
3472              **      D1 @ top of the string
3473              **      (MTHSTK) @ end of the string (beyond last character)
3474              **
3475              ** Exit: String header will be written on top of the string.
3476              **       D1 @ string header.
3477              **       (MTHSTK) @ string header.
3478              **       If not enough memory to generate the header(16 nibs),
3479              **       it will direct exit to MFERR error routine.
3480              **
3481              ** Calls: STK16?
3482              **
3483              ** Uses: A,B,C(A),D0,D1
3484              **
3485              ** Stk lvls: +1
3486              **
3487              ************************************************************
3488              ************************************************************
3489              *
3490 14C2E 1B99 =STRHED DO=(5) =MTHSTK
          5F2
3491 14C35 AF0          A=0    W
3492 14C38 142          A=DATO A
3493 14C3B 137          CD1EX
3494 14C3E 135  =STHD10 D1=C                  D1 @ TOP OF STACK
3495 14C41 EA           A=A-C  A              A= STRING KENGTH
3496 14C43 AF8          B=A    W
3497 14C46 8E76         GOSUBL STK16?
```

```
            1F
3498 14C4C 480          GOC    STHD20
3499 14C4F 8CB7         GOLONG PRT185
            7F
3500 14C55 31F0  STHD20 LCHEX  0F
3501 14C59 14D          DAT1=C B
3502 14C5C 171          D1=D1+ 2
3503 14C5F AF4          A=B    W
3504 14C62 159D         DAT1=A 14
3505 14C66 1C1          D1=D1- 2
3506 14C69 6A2D         GOTO   upmths
3507             *
3508             *
3509 14C6D 181   R&SKP+ D0=D0- 2
3510 14C70 14E          C=DAT0 B
3511 14C73 96A   R&SKP- ?C=0   B
3512 14C76 00           RTNYES
3513 14C78 1C1          D1=D1- 2
3514 14C7B 14D          DAT1=C B
3515 14C7E 02           RTNSC
3516             *
3517 14C80              END
```

```
 A-MULT  Ext                  -  2801
 ARYELM  Ext                  -  2809
 ASARRY  Abs   83127 #144B7   -  2406  2393
 ASC100  Abs   83100 #1449C   -  2392  2388
 ASC230  Abs   83114 #144AA   -  2400  2422
 ASC240  Abs   83164 #144DC   -  2422  2467
 ASCIIF  Abs   83076 #14484   -  2383  2318
 ASCN50  Abs   83090 #14492   -  2390  2395  2415  2417
 ASCSEC  Abs   83085 #1448D   -  2387  2384
 ASCSTR  Abs   83110 #144A6   -  2399
 AVMEMS  Abs  193940 #2F594   -    15  1220
 Array   Abs       1 #00001   -    23  1087  1093  1641  1715  1763  2362  2392
                              2495  2775
=BACK1B  Abs   80652 #13B0C   -   767   161  2208
=BACK2B  Abs   80650 #13B0A   -   766   240   846   868  2230
=BACK3B  Abs   80648 #13B08   -   765  1821  1837  1852
 BACKUP  Abs   80688 #13B30   -   784   778
 BADTYP  Abs   82983 #14427   -  2333  1625  1755
 BSERR   Ext                  -  2153
=BsErr   Abs   82608 #142B0   -  2153
 CHKCOD  Abs   81283 #13D83   -  1142  1587  2144
=CHKDON  Abs   81458 #13E32   -  1301  1089  1601  2305
=CHKEOL  Abs   81261 #13D6D   -  1131
 CHKRSP  Ext                  -   693
=CHKSTK  Abs   81334 #13DB6   -  1219
 CHN#SV  Abs  194927 #2F96F   -    15  1270
 CKBF10  Abs   80672 #13B20   -   777   782
 CKBPOS  Abs   80663 #13B17   -   774   770
 CKMFLG  Abs   81404 #13DFC   -  1264  1598  1712  2302
 CKST10  Abs   80654 #13B0E   -   769
 Check   Abs       7 #00007   -    30  2419  2423  2536  2540  2988
 Cmplex  Abs       3 #00003   -    25  1672  1683  1735  1890  1897  2414  2770
                              3081
=DO+2RD  Abs   80434 #13A32   -   516   336   888   893   948  1816  1861  1974
 DO+2RX  Abs   80457 #13A49   -   524
 DO+2WR  Abs   80502 #13A76   -   585    91  2478
 DO+2WX  Abs   80536 #13A98   -   598   592
=DO=FIB  Abs   80581 #13AC5   -   645   726   788  1010  2160  2580
=D1MST+  Abs   81441 #13E21   -  1273  1267
 D1MSTK  Ext                  -  2547
 D1UPMT  Abs   81371 #13DDB   -  1241  1253
=D1mstk  Abs   83452 #145FC   -  2547  1273  1629  1634  2756  3417
 DATB05  Abs   82429 #141FD   -  1934
 DATB10  Abs   82446 #1420E   -  1940  1963  1966  1968
 DATB15  Abs   82450 #14212   -  1941  1955
 DATB20  Abs   82458 #1421A   -  1945  1937
 DATB25  Abs   82469 #14225   -  1951  1947
 DATB30  Abs   82482 #14232   -  1961  1939  1952
 DATB55  Abs   82504 #14248   -  1972  1975
 DATB60  Abs   82519 #14257   -  1977  1973
 DATBEG  Abs   82423 #141F7   -  1932  1746  1768  1799  1982
 DATBER  Abs   82474 #1422A   -  1954  1871  1881
=DATCOD  Abs   84026 #1483A   -  2893  2897
 DATLEN  Ext                  -  2902  3032
 DEST    Ext                  -  1344
```

```
 DROPO2  Abs   81354 #13DCA -  1233  1727
 DROPST  Abs   81357 #13DCD -  1235   216  1659  1797  2523
 Done    Abs       6 #00006 -    29    99   117   126  1828  1831  1865
 EOFERX  Abs   83006 #1443E -  2349  2400
=EOFIL   Abs   81898 #13FEA -  1694   939  1609  1722  1729  1838  2713
 EOFILX  Abs   81904 #13FF0 -  1702  2225
 EOFLC+  Ext         ----- -  3317
 EOLCHK  Abs   81255 #13D67 -  1129  1090  1096  1103  1592  2181  2236
 EORC20  Abs   81188 #13D24 -  1087  1083  1758  2489
 EORC30  Abs   81219 #13D43 -  1098  1094
 EORCHK  Abs   81148 #13CFC -  1073  1879  2531
 ERREX   Abs   81908 #13FF4 -  1703  1750  1823  2179  2291  2460
 EXPEX-  Ext         ----- -  1342  2295
 EXPSKP  Ext         ----- -  2649
 Error   Abs       5 #00005 -    27   970  1623  1632  1652  1663  1666  1942
                             2191  2288  2349  2374  2412  3071
 F#NFND  Ext         ----- -  2591
 F-R1-3  Abs  194746 #2F8BA -    15  2794
 FDFBF1  Abs   83501 #1462D -  2593  2589
 FIBADR  Ext         ----- -  1272
 FIX100  Abs   83210 #1450A -  2446  2437
 FIX105  Abs   83241 #14529 -  2462  2458
 FIX110  Abs   83243 #1452B -  2463  2452
 FIX115  Abs   83250 #14532 -  2466  2464
 FIX118  Abs   83254 #14536 -  2467  2481
 FIX125  Abs   83279 #1454F -  2476  2479
 FIX140  Abs   83294 #1455E -  2481  2522
 FIX145  Abs   83297 #14561 -  2482  2472  2477
 FIX150  Abs   83322 #1457A -  2487  2490
 FIX160  Abs   83326 #1457E -  2489  2484
 FIX163  Abs   83332 #14584 -  2490  2533
 FIX165  Abs   83335 #14587 -  2491  2527
 FIX300  Abs   83338 #1458A -  2495  2438
 FIX350  Abs   83384 #145B8 -  2522  2539
 FIX360  Abs   83399 #145C7 -  2526  2545
 FIXARY  Abs   83359 #1459F -  2506  2496
 FIXLEN  Abs   83201 #14501 -  2436  2327
 FIXSPC  Ext         ----- -  1996
 FIXSTR  Abs   83368 #145A8 -  2517  2498
=FNDFBF  Abs   83458 #14602 -  2580  2661
 FORSTK  Abs  193950 #2F59E -    15  1100  1301
 FPOLL   Ext         ----- -   611
 FTYPER  Ext         ----- -  2157
 FXSA10  Abs   83425 #145E1 -  2538
 FXSP05  Abs   84466 #149F2 -  3140  3166
 FXSP10  Abs   84470 #149F6 -  3141  3139
 FXSP20  Abs   84496 #14A10 -  3152  3144
 FXSP30  Abs   84503 #14A17 -  3171
 FXSP40  Abs   84508 #14A1C -  3173  3179
 FXSP50  Abs   84524 #14A2C -  3180  3174
 FXSP70  Abs   84530 #14A32 -  3183  3147
 FXSP80  Abs   84537 #14A39 -  3186  3184
 FXSP90  Abs   82537 #14269 -  1985  3189
 FXSTAR  Abs   83418 #145DA -  2536  2507
 FXSTSP  Abs   84450 #149E2 -  3134  2521  2538
```

Fresh	Abs	8	#00008	-	32	1088	1275	1798	1801	1808	1812	1855
					2776	2911						
GETCH#	Ext			-	1583	2140						
GETRE#	Ext			-	1584	2141						
GETS10	Abs	84591	#14A6F	-	3296	3187	3294					
GETS12	Abs	84714	#14AEA	-	3339	3303						
GETS15	Abs	84737	#14B01	-	3345	3337						
GETS40	Abs	84928	#14BC0	-	3405	3403						
GETS50	Abs	84965	#14BE5	-	3417	3388						
GETS60	Abs	84980	#14BF4	-	3422	3420						
=GETSPC	Abs	84584	#14A68	-	3293	2201	2221	2347	2466			
GTPTR0	Abs	83562	#1466A	-	2659	2656						
GTPTR2	Abs	83675	#146DB	-	2697	2681						
GTPTR3	Abs	83685	#146E5	-	2701	2695						
GTPTR4	Abs	83714	#14702	-	2713	2705						
=GTPTRS	Abs	83510	#14636	-	2643	1144	1606	2190	2258	2312		
=GTPTRX	Abs	83568	#14670	-	2661							
H41D10	Abs	82989	#1442D	-	2339	2332						
H41D20	Abs	83021	#1444D	-	2356	2348						
H41D40	Abs	83041	#14461	-	2362	2345						
H41D45	Abs	83049	#14469	-	2368	2391						
H41D50	Abs	83056	#14470	-	2373	2363	2511					
HP41DT	Abs	82978	#14422	-	2331	2326						
I/OFND	Ext			-	2588							
I/Obuf	Abs	10	#0000A	-	34	517	586	769	1018	2223	2270	2351
					2463	2585	2651	2680	3183	3296		
IDIV	Ext			-	1045	3400						
Iram	Abs	11	#0000B	-	35	2652	2671	3302				
LDC=8	Abs	82546	#14272	-	1991	2470						
LDEO	Abs	82568	#14288	-	2001	1682	1899					
LDEO-	Abs	82565	#14285	-	2000	2416						
LOOPBK	Abs	82935	#143F7	-	2305	2360	2402	2432	2487			
LSTADR	Ext			-	3322							
=MFLG=0	Abs	81313	#13DA1	-	1186	1143						
=MFLG=X	Abs	81315	#13DA3	-	1187							
MLFFLG	Abs	194672	#2F870	-	15	1187	1264					
MOVED3	Ext			-	3376							
MTHSTK	Abs	193945	#2F599	-	15	1241	2978	3490				
MfErr	Ext			-	1703							
N-ST10	Abs	85002	#14C0A	-	3447	3450						
N-ST20	Abs	85014	#14C16	-	3452	3448						
N-ST30	Abs	85034	#14C2A	-	3458							
N-STR	Abs	84988	#14BFC	-	3442	1665						
NEWITM	Abs	83799	#14757	-	2784	2777						
NEWVAR	Ext			-	1350							
NEXTst	Ext			-	2233							
NOMEM	Ext			-	2292							
NOSPC	Abs	83065	#14479	-	2377	2352	2413					
NOSPC2	Abs	83069	#1447D	-	2378	2354						
NOTIRM	Abs	83604	#14694	-	2672	2670						
NXTA10	Abs	84032	#14840	-	2894	2872						
NXTA20	Abs	84086	#14876	-	2910	2892						
NXTA30	Abs	84097	#14881	-	2914	2912						
=NXTADR	Abs	83944	#147E8	-	2866	1671	1733	1886				
NXTE02	Abs	84169	#148C9	-	2999	2995						

```
 NXTE05  Abs   84171 #148CB -  3000  2989
 NXTE12  Abs   84217 #148F9 -  3016  3010
 NXTE15  Abs   84245 #14915 -  3028  3015
 NXTE20  Abs   84250 #1491A -  3030  3004
 NXTE25  Abs   84285 #1493D -  3041  3037
 NXTE30  Abs   84307 #14953 -  3050  3085  3089
 NXTE40  Abs   84353 #14981 -  3065  3040  3075  3079  3091
 NXTE50  Abs   84398 #149AE -  3078  3093
 NXTE60  Abs   84405 #149B5 -  3080  3077
 NXTE70  Abs   84435 #149D3 -  3090  3083
 NXTECK  Abs   84110 #1488E -  2978  2866  2987
=NXTELM  Abs   84140 #148AC -  2987  2373  2411  2420  2424  2537  2541
 NXTLST  Abs   82942 #143FE -  2307  2311
 NXTNOR  Abs   84378 #1499A -  3071  3067
 NXTREL  Abs   84393 #149A9 -  3076  3048
 NXTSHT  Abs   84383 #1499F -  3073  3056
 NXTV10  Abs   81532 #13E7C -  1353  1351
 NXTV20  Abs   81574 #13EA6 -  1366  1360
 NXTV30  Abs   81589 #13EB5 -  1370  1365
=NXTVA-  Abs   81496 #13E58 -  1343
=NXTVAR  Abs   81484 #13E4C -  1341  1597
=NXtstm  Abs   82820 #14384 -  2232
 Notnum  Abs       0 #00000 -    22  1620  1689  1743  2344  2387  2436  2761
 Odd     Abs       6 #00006 -    28   234   858   891  3236
 PCADDR  Abs  194169 #2F679 -    15  2645
 POLL    Ext                -  1280
 POPMTH  Ext                -  1098  1635
=Polljj  Abs   81452 #13E2C -  1279
=PRINT#  Abs   82577 #14291 -  2139
 PRT100  Abs   82605 #142AD -  2149  1591
 PRT105  Abs   82615 #142B7 -  2155  2149
 PRT106  Abs   82626 #142C2 -  2158  2156
 PRT110  Abs   82630 #142C6 -  2160  2145
 PRT120  Abs   82662 #142E6 -  2170  2166
 PRT121  Abs   82689 #14301 -  2180  2173  2175  2177
 PRT123  Abs   82706 #14312 -  2190  2182
 PRT124  Abs   82729 #14329 -  2201  2286
 PRT125  Abs   82749 #1433D -  2207  2205
 PRT130  Abs   82765 #1434D -  2211  2197
 PRT131  Abs   82780 #1435C -  2220  2217
 PRT135  Abs   82786 #14362 -  2222  2202
 PRT136  Abs   82798 #1436E -  2227  2222
 PRT140  Abs   82820 #14384 -  2233  1594  2158  2212  2289
 PRT150  Abs   82826 #1438A -  2236  2307
 PRT170  Abs   82833 #14391 -  2258  2379
 PRT175  Abs   82863 #143AF -  2273  2266
 PRT178  Abs   82876 #143BC -  2282  2278
 PRT180  Abs   82880 #143C0 -  2288  2209  2261  2271  2280
 PRT185  Abs   82892 #143CC -  2292  1352  1678  1854  1903  2224  3499
 PRT200  Abs   82898 #143D2 -  2294  2183  2237
 PRT201  Abs   82927 #143EF -  2302  2299
 PRT210  Abs   82946 #14402 -  2309  2306  2369  2375  2504
 PRT215  Abs   82948 #14404 -  2310  2303
 PRT220  Abs   82955 #1440B -  2312  3428
 PRT230  Abs   82968 #14418 -  2325
```

```
 PRTERR  Abs   82685 #142FD -  2179  2168
 PRTPOL  Abs   82599 #142A7 -  2147
 R&SKP+  Abs   85101 #14C6D -  3509  3449  3458
 R&SKP-  Abs   85107 #14C73 -  3511  3457
 R41C10  Abs   81712 #13F30 -  1622  1685
 R41C15  Abs   81716 #13F34 -  1623
 R41C25  Abs   81729 #13F41 -  1630  1667  1731
 R41C26  Abs   81744 #13F50 -  1634  1713
 R41C30  Abs   81766 #13F66 -  1641  1621
 R41C35  Abs   81776 #13F70 -  1648  1684
 R41C40  Abs   81798 #13F86 -  1658  1644
 R41C45  Abs   81808 #13F90 -  1661  1680
 R41C50  Abs   81832 #13FA8 -  1671  1642
 R41C55  Abs   81849 #13FB9 -  1676  1673
 R41C60  Abs   81860 #13FC4 -  1679  1677
 R41CER  Abs   81762 #13F62 -  1639  1633
 RASC10  Abs   81891 #13FE3 -  1692  1739
 RASC20  Abs   81914 #13FFA -  1704  1693  1723
 RASC30  Abs   81949 #1401D -  1715  1690
 RASC35  Abs   81963 #1402B -  1721  1736
 RASC40  Abs   81973 #14035 -  1727  1718
 RASC45  Abs   81977 #14039 -  1728  1738
 RASC50  Abs   81992 #14048 -  1733  1716
=RCLNUM  Abs   84296 #14948 -  3045
 RDATTY  Ext                -  2333
=RDBYTA  Abs   80431 #13A2F -   515   375   378   384   387
=RDLNAS  Abs   80415 #13A1F -   383   840
=RDLNFX  Abs   80391 #13A07 -   374  1817
 RDNUM0  Abs   80905 #13C09 -   931
 RDNUM1  Abs   80919 #13C17 -   942   938
 RDNUM2  Abs   80922 #13C1A -   943   949
 RDNUM5  Abs   80945 #13C31 -   953   944
=READ#   Abs   81595 #13EBB -  1582
 READIN  Ext                -  3060
 RECADR  Ext                -  1369
 RECERR  Abs   83233 #14521 -  2459  1760  2455  2491  3140
 RED150  Abs   81627 #13EDB -  1592  1588  1602  1605
 RED200  Abs   81638 #13EE6 -  1596  1593
 RED230  Abs   81653 #13EF5 -  1601  1637
 RED235  Abs   81657 #13EF9 -  1602
 RED236  Abs   81662 #13EFE -  1604  1599
 RED238  Abs   81684 #13F14 -  1610  1608
 RED240  Abs   81697 #13F21 -  1614  1612
 RED41C  Abs   81707 #13F2B -  1620  1615
 REDASC  Abs   81882 #13FDA -  1689  1613
 REDFIX  Abs   82012 #1405C -  1743  1616
=REDNUM  Abs   80899 #13C03 -   924  1622  1649  1651  1661  1756  1774
=REDREC  Abs   80747 #13B6B -   838  1692  1721  1728
 REDSTO  Abs   81725 #13F3D -  1629  1624  1654  1761  1882
 RESTDO  Ext                -  1129
 RESTRO  Abs   80525 #13A8D -   593   523
 RFAD+I  Ext                -  3380
 RFUPD+  Ext                -  3382  3384
 RFX040  Abs   82017 #14061 -  1746  1787  1901
 RFX050  Abs   82031 #1406F -  1752  1747
```

```
 RFX100  Abs   82034 #14072 -  1753
 RFX110  Abs   82039 #14077 -  1755  1772  1809
 RFX120  Abs   82043 #1407B -  1756  1754
 RFX125  Abs   82061 #1408D -  1761  1757  1759
 RFX200  Abs   82065 #14091 -  1763  1744
 RFX210  Abs   82075 #1409B -  1768  1900
 RFX220  Abs   82110 #140BE -  1788  1764
 RFX250  Abs   82114 #140C2 -  1792  1766
 RFX260  Abs   82133 #140D5 -  1798  1889
 RFX270  Abs   82136 #140D8 -  1799  1873
 RFX280  Abs   82152 #140E8 -  1805  1800
 RFX290  Abs   82161 #140F1 -  1808  1815
 RFX310  Abs   82172 #140FC -  1812  1807
 RFX320  Abs   82182 #14106 -  1816  1813
 RFX330  Abs   82210 #14122 -  1825  1820
 RFX340  Abs   82230 #14136 -  1834  1830
 RFX350  Abs   82245 #14145 -  1840  1836
 RFX355  Abs   82281 #14169 -  1854
 RFX360  Abs   82285 #1416D -  1855  1851
 RFX370  Abs   82288 #14170 -  1856  1862
 RFX380  Abs   82310 #14186 -  1864  1857
 RFX382  Abs   82329 #14199 -  1871  1811
 RFX385  Abs   82336 #141A0 -  1873  1870
 RFX390  Abs   82340 #141A4 -  1878  1803  1866  1872
 RFX395  Abs   82357 #141B5 -  1882  1880
 RFX420  Abs   82374 #141C6 -  1890  1888
 RFX430  Abs   82385 #141D1 -  1894  1891
 RFX435  Abs   82409 #141E9 -  1901  1898
 RFX440  Abs   82413 #141ED -  1903  1895
 RFXARY  Abs   82361 #141B9 -  1886  1788
 RFXERR  Abs   82024 #14068 -  1749  1639  1769  1802
 ROMF-   Ext                -  3314
 RREC20  Abs   80773 #13B85 -   850   844
 RREC25  Abs   80798 #13B9E -   860   857
 RREC30  Abs   80827 #13BBB -   871   866
 RREC40  Abs   80861 #13BDD -   883   889
 RREC50  Abs   80883 #13BF3 -   890   884
 RREC60  Abs   80895 #13BFF -   894   892
 RSREGS  Abs   80596 #13AD4 -   681   529   608
 Random  Abs       9 #00009 -    33    87   103  1058  1073  1869  1938  1951
                              2204  2216  2277  2281  2383  2454  2650  2657
                              3165
 Return  Abs       4 #00004 -    26  2192  2262  3419
 S-R0-0  Abs  194673 #2F871 -    15  2815  2875
 S-R0-3  Abs  194688 #2F880 -    15  1353  2904
 S-R1-0  Abs  194689 #2F881 -    15  3304
 S-R1-1  Abs  194694 #2F886 -    15  3320  3343  3379
 SAVEDO  Ext                -  1906
=SETWRT  Abs   80632 #13AF8 -   726   241   605  2180  2486
=SKPBYT  Abs   80378 #139FA -   334   337
 SNAPSV  Ext                -   610
=STHD10  Abs   85054 #14C3E -  3494
 STHD20  Abs   85077 #14C55 -  3500  3498
=STK16?  Abs   81331 #13DB3 -  1218  1676  1894  3066  3497
=STK19?  Abs   81328 #13DB0 -  1217
```

```
 STKSPC  Abs   81337 #13DB9 -  1220   865  1848
 STKV10  Abs   83764 #14734 -  2771  2765  2769
 STKV15  Abs   83783 #14747 -  2778  1105
 STKV20  Abs   83830 #14776 -  2794  2792
 STKV25  Abs   83863 #14797 -  2803  2797  2799
 STKV40  Abs   83873 #147A1 -  2807  2786
 STKV45  Abs   83885 #147AD -  2810  2830  2832
 STKV50  Abs   83889 #147B1 -  2812  2805
 STKV60  Abs   83917 #147CD -  2823  2808
 STKV70  Abs   83934 #147DE -  2829  2828
=STKVC+  Abs   83720 #14708 -  2756
=STKVCT  Abs   83724 #1470C -  2757  1092  1604  2310
 STMTD0  Abs  194705 #2F891 -    15  3381
 STMTD1  Abs  194710 #2F896 -    15   645  1075  3306  3345  3409
 STORE   Ext                -  1631  1706
=STORTN  Abs   81941 #14015 -  1712  1704
 STR$SB  Ext                -  2390
 STRARY  Abs   83153 #144D1 -  2419  2407
=STRHED  Abs   85038 #14C2E -  3490  1878
=STROVF  Abs   82202 #1411A -  1822
 STRS10  Abs   84577 #14A61 -  3238  3235
=STRSPC  Abs   84548 #14A44 -  3222  2399
 STSPC-  Abs   84560 #14A50 -  3232  2421
 SVTRC   Ext                -  1341
 SYSFLG  Abs  194265 #2F6D9 -    15  2784
 String  Abs       2 #00002 -    24  1643  1679  1717  1737  1765  1887  2331
                              2394  2406  2497  2506  2762  2766  2807  2871
                              3003
 UCP010  Abs   81048 #13C98 -  1024  1019
 UCP020  Abs   81069 #13CAD -  1032  1022
 UCP025  Abs   81086 #13CBE -  1039  1034
 UCP030  Abs   81121 #13CE1 -  1051  1049
 UCP040  Abs   81146 #13CFA -  1060  1057  1074
=UPCPOS  Abs   80999 #13C67 -  1007   154   237   302   894   964   969  1864
 UPMH16  Abs   81392 #13DF0 -  1250  1691
 UPMH34  Abs   81386 #13DEA -  1248  1720
 UPMTHS  Abs   81397 #13DF5 -  1252   301  1636  1681  1730  1896  2368  2503
                              3070
 UpMstk  Abs       7 #00007 -    31
 VAL00   Ext                -  1710
 ValSub  Ext                -  1709
=WRBYTC  Abs   80499 #13A73 -   584   106   108   112   122   160   225   227
                               232   236   576   580
=WRBYTD  Abs   80493 #13A6D -   582   298
=WRSTR*  Abs   80243 #13973 -   217  2431
 WRTEOF  Abs   80475 #13A5B -   575   238   239  2229
 WRTMRK  Abs   80483 #13A63 -   578  2207  2228
 WRTN10  Abs   80360 #139E8 -   298   300
=WRTNUM  Abs   80324 #139C4 -   286  2359  2483
=WRTSTR  Abs   80239 #1396F -   216  2401
 WSTF05  Abs   80074 #138CA -    89    92
 WSTF08  Abs   80087 #138D7 -    94    90
 WSTF10  Abs   80097 #138E1 -    98    86
 WSTF30  Abs   80120 #138F8 -   106   102   135   137
 WSTF40  Abs   80154 #1391A -   118   114   123
```

```
 WSTF50   Abs   80173 #13920 -   124   119
 WSTF70   Abs   80211 #13953 -   154   125   127
 WSTF80   Abs   80225 #13961 -   159   157  2485
 WSTF90   Abs   80237 #1396D -   162   155
 WSTR20   Abs   80274 #13992 -   228   233
 WSTR30   Abs   80293 #139A5 -   234   231
 WSTR40   Abs   80302 #139AE -   237   235
=WSTRFX   Abs   80053 #138B5 -    81  2526
 badtyp   Abs   81721 #13F39 -  1625  1653  1664
 chkspc   Ext                -  3339
 eEOFIL   Ext                -  1702  1940  2350
 eFACCS   Ext                -  2178
 eFPROT   Ext                -  2167
 eMEM     Ext                -  2377
 eRECOR   Ext                -  1954  2459
 eSTROV   Ext                -  1822
 fAOS     Abs     223 #000DF -    14   100
 fEOF     Abs     255 #000FF -    14   575  1935  2203  2227
 fEOR     Abs     239 #000EF -    14   159  1945  2206
 fEOS     Abs     111 #0006F -    14   133
 fMOS     Abs     127 #0007F -    14   136
 fSOS     Abs     207 #000CF -    14   105
=fpoll    Abs   80575 #13ABF -   611  1694
 fpoll+   Abs   80568 #13AB8 -   610   527   601   804
 ftyper   Abs   82620 #142BC -  2157  2218  2385
 oACCSb   Abs      11 #0000B -    14   727
 oCOPYb   Abs      10 #0000A -    14  1011  1013  2662  2665  3348  3350
 oCPOSb   Abs      40 #00028 -    14   790  1016  1051  2679  2701
 oDBEGb   Abs      21 #00015 -    14  1013  1015  2672  2675  3309  3359  3360
 oDEVCb   Abs      12 #0000C -    14  2665  2672
 oDLENb   Abs      46 #0002E -    14  2701  2708  3361
 oFBEGb   Abs      13 #0000D -    14  3350  3359
 oFBF#b   Abs       2 #00002 -    14  2581  2662
 oFLENh   Abs      32 #00020 -    14  3354
 oFTYPh   Abs      16 #00010 -    14  3353  3354
 oPROTb   Abs       9 #00009 -    14  2161
 oREC#b   Abs      32 #00020 -    14  3360  3361  3411
 oRECLb   Abs      36 #00024 -    14  1015  1016  2675  2679
 oRLENb   Abs      52 #00034 -    14  1051  1077  2708
 pEOFIL   Abs      37 #00025 -    14  1695
 pPRIN#   Abs      38 #00026 -    14  2148
 pRDCBF   Abs      24 #00018 -    14   805
 pRDNBF   Abs      25 #00019 -    14   528   602
 pREAD#   Abs      39 #00027 -    14  1590
 poll     Abs   81452 #13E2C -  1280  1589  2147
 prt185   Abs   81528 #13E78 -  1352   869
 saved0   Abs   82417 #141F1 -  1906  1142  1343  2296
 tCOMMA   Ext                -  1132
 tSEMIC   Ext                -  2654
 upnths   Abs   84372 #14994 -  3070  3506
```

Input Parameters

  Source file name is SC&DAT::MS

  Listing file name is SC/DAT:TI:ML::-1

  Object file name is SC%DAT:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    M   M  N   N   &     EEEEE  DDDD
 2            *    MM MM  N   N  & &    E      D  D
 3            *    M M M  NN  N  & &    E      D  D
 4            *    M M M  N N N   &     EEEE   D  D
 5            *    M   M  N  NN  & & &  E      D  D
 6            *    M   M  N   N  &  &   E      D  D
 7            *    M   M  N   N   && &  EEEEE  DDDD
 8            *
 9                      TITLE  Character Editor <831213.1314>
10 14C80               ABS    #14C80
11            *
12             Insert EQU    7
13             BitsOK EQU    1
14             fTMOUT EQU    3
```

```
                    STITLE CHEDIT
               *
14C80 B125  RCURON NIBHEX B125B1E3FF     Replace cursor, Cursor on
      B1E3
      FF
               *
               ********************************************************************
               ********************************************************************
               **
               ** Name:(S) NOSCRL  -  Request No-display-scrolling
               **
               ** Category:   DSPUTL
               **
               ** Purpose:
               **      Request that main loop bypass scrolling of current
               **      display contents.
               **
               ** Entry:
               **      None.
               **
               ** Exit:
               **      C[A]=0.
               **      DO=NEEDSC.
               **
               ** Calls:      None.
               **
               ** Uses.......
               **              C[A],DO.
               **
               ** Stk lvls:   0
               **
               ** Detail:
               **      Clears (NEEDSC).  This prevents main loop from calling
               **      SCRLLR so user can stare at display.
               **
               ** History:
               **
               **    Date      Programmer              Modification
               **   --------   ----------   ----------------------------------
               **   10/31/83   NM           Added documentation
               **
               ********************************************************************
               ********************************************************************
14C8A          =NOSCRL
14C8A 1B00   CHEDEX DO=(5) =NEEDSC
      000
14C91 D2             C=0    A
14C93 15C0           DAT0=C 1              Don't need to scroll
14C97 01             RTN
               ****************************************************************
               ****************************************************************
               **
               ** Name:(S) CHEDIT  -  Character Editor
               **
               ** Category:   KEYUTL
```

```
67          **
68          ** Purpose:
69          **      Accepts keyboard input and edits line in display.
70          **
71          ** Entry:
72          **      P=0, Hexmode
73          **
74          ** Exit:
75          **      P=0
76          **      If carry set then A(A)=Function code.
77          **      If carry clear then CHEDIT was terminated by an
78          **        immediate execute key.  R3(A)=Definition length.
79          **        D1 points to first char of definition.
80          **
81          **
82          ** Calls:      CHEDEX, CHROUT, DSPCHA, DSPCHR, DSPCL?, DSPSPC,
83          **             KEYRD, TBLJMC, WRIT05, WRITE, bf2dsp.
84          **
85          ** Uses:
86          **      A,B,C,D,P,D0,D1,R0,R3,ST,DEFADR,USRSTA,32 nibs at
87          **      SCRTCH.
88          **
89          ** Stk lvls:   6
90          **
91          ** Detail:
92          **      This subroutine implements a character editor which
93          **      accepts keyboard input and edits display as needed
94          **      until a key is entered which is not meaningful in
95          **      character edit mode.  The keycode of the terminator
96          **      is returned in the A register.  The following keys
97          **      are terminators:
98          **
99          **      A(A)  Key#   Function
100         **       13 -- 38 -- EndLine
101         **       14 -- 43 -- Attention
102         **       15 -- 46 -- RUN key
103         **       16 --112 -- CONTinue key
104         **       17 --102 -- SST key
105         **       18 -- 50 -- Cursor up
106         **       19 -- 51 -- Cursor down
107         **       20 --162 -- Cursor to top
108         **       21 --163 -- Cursor to bottom
109         **       22 --155 -- g Attention
110         **       23 --111 -- CALC mode key
111         **       24 -- 99 -- OFF key
112         **       25 --164 -- g EndLine (Cmd Stack)
113         **
114         **      Although these keycodes map to the same values as
115         **      certain control keys (ctrl-M through ctrl-Y), hitting
116         **      the CTRL sequence followed by a key will NOT be
117         **      interpreted as one of these terminators with the
118         **      exception of CTRL-M.  They will simply be put into the
119         **      display as funny-looking characters.
120         **
121         ** History:
```

```
122                **
123                **    Date      Programmer          Modification
124                **  --------    ----------  ------------------------------
125                **  06/23/82    BS          Updated documentation
126                **  11/05/82    NM          Rewrote
127                **
128                ****************************************************************
129                ****************************************************************
130 14C99 1F08 =CHEDIT D1=(5) RCURON
          C41
131 14CA0 7295         GOSUB  bf2dsp        Turn on replace cursor
132 14CA4 8F00         GOSBVL =DSPCL?
          000
133 14CAB 7261  CHED00 GOSUB  KEYRD         Read a key
134 14CAF 1B00         D0=(5) =DEFADR
          000
135 14CB6 15E7         C=DAT0 8             Read key definition
136 14CBA 0A           ST=C                 Key type to S:11-8
137 14CBC 10B          R3=C                 Definition length to R3[B]
138 14CBF BF6          CSR    W
139 14CC2 BF6          CSR    W
140 14CC5 816          CSRC                 Key type to C[S]
141 14CC8 135          D1=C                 Text address to D1
142 14CCB 86B          ?ST=0  11            Lex item?
143 14CCE 92           GOYES  CHED50        No
144 14CD0 86A          ?ST=0  10            Yes.  Leading space?
145 14CD3 60           GOYES  CHED10        No
146 14CD5 7601         GOSUB  DSPSPC        Yes, output space
147 14CD9 71C0  CHED10 GOSUB  WRITE         Write text.  Won't contain CR.
148 14CDD 869          ?ST=0  9             Trailing space?
149 14CE0 60           GOYES  CHED30        No.
150 14CE2 79F0         GOSUB  DSPSPC        Yes.  Output space.
151 14CE6 868   CHED30 ?ST=0  8             Paren needed?
152 14CE9 2C           GOYES  CHED00        No.
153 14CEB 3182         LCASC  \(\           Yes
154 14CEF 70F0         GOSUB  DSPCHR
155 14CF3 67BF  CHED40 GOTO   CHED00        Back for more keys.
156 14CF7 868   CHED50 ?ST=0  8             CTRL character?
157 14CFA 52           GOYES  CHED60        No.
158 14CFC D2           C=0    A             Yes.
159 14CFE 10B          R3=C                 Zero character counter
160 14D01 14B          A=DAT1 B             Read char
161 14D04 3104         LC(2)  64
162 14D08 EA           A=A-C  A             Convert to CTRL char.
163 14D0A 75B0         GOSUB  WRIT05        Write it.
164 14D0E 44E          GOC    CHED40        Go if not CR.
165 14D11 D2    RTNCR  C=0    A             Terminate CHEDIT with CR.
166 14D13 31D0         LC(2)  13
167 14D17 DA           A=C    A
168 14D19 7D6F  CHEDEj GOSUB  CHEDEX
169 14D1D 02           RTNSC
170 14D1F 94A   CHED60 ?C=0   S             ASCII?
171 14D22 F2           GOYES  CHED70        Yes.
172 14D24 86A          ?ST=0  10            No.  This is UDK.  Type 4,6?
173 14D27 A1           GOYES  UDK;          No.
```

```
174 14D29 869          ?ST=0  9            Yes.  Type 6?
175 14D2C 51           GOYES  UDK;         No.
176 14D2E 11B          C=R3                Yes.  Get length.
177 14D31 D0           A=0    A
178 14D33 AEA          A=C    B            Text len in bytes.
179 14D36 C4           A=A+A  A
180 14D38 E4           A=A+1  A            Text len + 1 in nibs.
181 14D3A 103          R3=A
182 14D3D 6C4F         GOTO   CHEDEX       Exit CHEDIT with carry clear.
183 14D41 7950  UDK;   GOSUB  WRITE        Write UDK.
184 14D45 5BC          GONC   RTNCR        Go if encountered CR.
185 14D48 86A          ?ST=0  10           Terminating (type 4)?
186 14D4B 6C           GOYES  RTNCR        Yes.
187 14D4D 6D5F         GOTO   CHED00       Back for more.
188 14D51 14B   CHED70 A=DAT1 B            Fetch ASCII char
189 14D54 31A0         LC(2)  ((TBEN)-(TBST))/3
190 14D58 9EE          ?A>=C  B            Special function?
191 14D5B 23           GOYES  CHED80       No.
192 14D5D 34BA         LC(5)  CHED00       Yes.
          C41
193 14D64 06           RSTK=C              Load RTN addr.
194 14D66 D6           C=A    A
195 14D68 8F00         GOSBVL =TBLJMC      And do special function
          000
196 14D6F E05   TBST   REL(3) -CHAR        0 = -char
197 14D72 000          CON(3) 0            1 = LC toggle (done in KEYRD)
198 14D75 0E4          REL(3) IN/REP       2 = In/Rep toggle
199 14D78 000          CON(3) 0            3 = User mode toggle (KEYRD)
200 14D7B AF4          REL(3) -LINE        4 = -line
201 14D7E 164          REL(3) CURSFL       5 = Cursor far left
202 14D81 654          REL(3) CURSFR       6 = Cursor far right
203 14D84 364          REL(3) BCKSPC       7 = Backspace
204 14D87 844          REL(3) CURSRL       8 = Cursor left
205 14D8A D34          REL(3) CURSRR       9 = Cursor right
206 14D8D       TBEN
207 14D8D 3102  CHED80 LC(2)  32
208 14D91 9E2          ?A<C   B            Is this a terminating keycode?
209 14D94 58           GOYES  CHEDEj       Yes.
210 14D96 7400         GOSUB  WRITE        No.  Write out char.
211 14D9A 601F         GOTO   CHED00       Back for more.
212             *
213 14D9E 11B   WRITE  C=R3
214 14DA1 A6E          C=C-1  B            Decrement character counter.
215 14DA4 400          RTNC                RTNSC if done.
216 14DA7 10B          R3=C
217 14DAA 1B00         DO=(5) (=DEFADR)+3
          000
218 14DB1 146          C=DAT0 A
219 14DB4 135          D1=C                Point at character
220 14DB7 14B          A=DAT1 B            Read character.
221 14DBA 171          D1=D1+ 2
222 14DBD 137          CD1EX
223 14DC0 144          DAT0=C A            Update pointer in DEFADR
224 14DC3 31A0  WRIT05 LC(2)  10
225 14DC7 962          ?A=C   B            LF?
```

```
226 14DCA 4D          GOYES  WRITE          Yes; ignore.
227 14DCC 31D0        LC(2)  13             No.
228 14DD0 962         ?A=C   B              CR?
229 14DD3 A0          GOYES  WRIT10         Yes, RTNCC.
230 14DD5 7C00        GOSUB  CHROUT         No.  Write character.
231 14DD9 64CF        GOTO   WRITE
232 14DDD 03   WRIT10 RTNCC
233             *
234 14DDF 3102 DSPSPC LCASC  \ \
235 14DE3 DA   DSPCHR A=C    A
236 14DE5 1B00 CHROUT DO=(5) =ESCSTA
          000
237 14DEC 146         C=DATO A
238 14DEF 108         RO=C                  Hold (ESCSTA),(FIRSTC),(CURSOR)
239 14DF2 8F00        GOSBVL =DSPCHA        Write character.
          000
240 14DF9 1B00        DO=(5) =ESCSTA
          000
241 14E00 142         A=DATO A              Read new (ESCSTA), etc.
242 14E03 118         C=RO                  Get old (ESCSTA), etc.
243 14E06 8A6         ?A#C   A              Anything changed?
244 14E09 00          RTNYES                Yes, on to next char.
245 14E0B 8C00        GOLONG =CHIRP         No, chirp first
          00
```

```
246                 STITLE Key-reading routine
247           ********************************************************************
248           ********************************************************************
249           **
250           ** Name:(S) KEYRD   -  Read A Key
251           **
252           ** Category:   KEYUTL
253           **
254           ** Purpose:
255           **      Read a key and return a pointer to its expanded value.
256           **
257           ** Entry:
258           **      HEX mode.
259           **      flRPTD and last position in keybuffer contain
260           **        information necessary for repeating keys to work.
261           **
262           ** Exit:
263           **      P=0.
264           **      DEFADR contains pointer to expanded value:
265           **        DEFADR: Length of string in bytes.
266           **        DEFADR+2: Key type:
267           **                    0 = Single ASCII character.  Includes
268           **                        control characters 0-31, which
269           **                        should usually cause some action
270           **                        in the editor calling KEYRD.
271           **
272           **                    1 = ASCII control character.  Must
273           **                        subtract #40 from the 1-byte def-
274           **                        inition we are pointing to.  These
275           **                        characters should be interpreted as
276           **                        text, and should not cause any
277           **                        special action in the editor.
278           **
279           **                    2 = User-defined key; Terminating.
280           **
281           **                    4 = User-defined key; Non-terminating.
282           **
283           **                    6 = User-defined key; Non-displaying.
284           **
285           **                    8-F = LEX entry with lower 3 bits as
286           **                          follows:
287           **                            bit 0: Parenthesis needed.
288           **                            bit 1: Trailing space needed.
289           **                            bit 2: Leading space needed.
290           **                          (spaces & paren not included in
291           **                           string length field)
292           **
293           **        DEFADR+3: Address of text.
294           **
295           ** Calls:      ALMSRV, ASLW5, BLDDSP, CSLW3, FINDAj, FLIP0,
296           **             FLIPCS, FPOLL, GETDEF, KEYTYP, MTADDR, POPBUF,
297           **             RPTKY, SETTMO, SFlagC, SLEEP, Sflag?, VWFC-2,
298           **             WIPOUT, cksreq, range, sflagt, usrsta.
299           **
300           ** Uses.......
```

```
301             **              A,B,C,D,P,D0,D1,R3,USRSTR (for holding ST),
302             **              DEFADR (for definition), 32 nibs at SCRTCH.
303             **
304             ** Stk lvls:   5
305             **
306             ** Algorithm:
307             **   KEYRD:  Build display.
308             **   KEYR50: Perform WTKY fastpoll.
309             **           If handled then goto KEYR69.
310             **           Check for repeating keys (RPTKY).
311             **           If we have a repeating key then goto KEYR72.
312             **           Build display.
313             **           Set 10-minute timeout (SETTMO).
314             **   KEYR60: Go to light sleep (SLEEP).
315             **           If key in buffer then goto KEYR70.
316             **           If 10-minute timeout not expired then goto KEYR60
317             **             else return OFF-key definition.
318             **   KEYR69: Set up registers after poll.
319             **           Goto KEYR72.
320             **   KEYR70: Pop key# from buffer.
321             **   KEYR72: Put key# and logical keycode in R0.
322             **           Perform KYDF fastpoll.
323             **           If handled then if S0=0 (not-returning-definition)
324             **             then goto KEYR50 else return.
325             **           If VIEW flag is clear then goto KEYR75.
326             **           Clear VIEW flag.
327             **           Get key definition; if none then goto VIEWUN.
328             **           Write definition to LCD.
329             **           Goto VIEW30.
330             **   VIEWUN: Write "Unassigned" to LCD.
331             **           Loop until keys up (VWFC-2).
332             **           Goto KEYR50.
333             **   KEYR75: If CTRL flag clear then goto KEYR80.
334             **           If keycode not in CTRL'able range then goto
335             **             KEYR80.
336             **           Return CTRL key definition.
337             **   KEYR80: If USRX flag clear then goto KEYR90.
338             **           Clear USRX flag.
339             **           Toggle USER flag (carry reflects old state).
340             **           Goto KEY100.
341             **   KEYR90: Carry := USER flag.
342             **   KEY100: If carry clear (not USER) then goto KEY110.
343             **           Fetch key redefinition.  If non-existent then
344             **             goto KEY110.
345             **           Return redefined key definition.
346             **   KEY110: RSTK=KEY120 (return address in case we do
347             **             internal processing).
348             **           {start of internal processing jump table}
349             **           If keycode = LC key then goto LOWERC.
350             **           If keycode = USER key then goto USERK.
351             **           If keycode = CTRL key then goto CTRL.
352             **           If keycode = VIEW key then goto VIEWK.
353             **           If keycode = temp-user key then goto USERX.
354             **           If keycode = Last-err key then goto lerrm.
355             **           {end of internal processing jump table}.
```

```
                **          Pop RSTK.
                **          If keycode in range of typing aids then goto
                **            NEWTOK.
                **          If LC flag set then flip case if appropriate
                **            (FLIPCS).
                **          {we have a simple 1-char definition}
                **          Look up key definition in KEYCOD table and return
                **            definition.
                **    KEY120: {we have finished internal processing}
                **          If keybuffer empty then zero out last entry in
                **            keybuffer to disable repeating key.
                **          Goto KEYR50.
                **    NEWTOK: Find typing aid definition (MTADDR).
                **       .  Return definition.
                **
                ** History:
                **
                **    Date      Programmer          Modification
                **  --------   ----------   ----------------------------------
                **  11/02/82   NM           Began to write.
                **
                ********************************************************************
                ********************************************************************
14E11 8F00 =KEYRD  GOSBVL =BLDDSP       Build display & store user status
      000
14E18 8E00  KEYR50 GOSUBL =FPOLL        Poll on entry to KEYRD
      00
14E1E 00           CON(2) =pWTKY
14E20 831          ?XM=0                Handled?
14E23 05           GOYES  KEYR69        Yes, key# is in R0[B].
14E25 07           C=RSTK
14E27 10B          R3=C                 Save a level
14E2A 7C84         GOSUB  RPTKY         Check for repeating keys.
14E2E 11B          C=R3
14E31 06           RSTK=C               Restore a level
14E33 445          GOC    KEYR72        Rebuild definition if rpt.
14E36 7F91         GOSUB  usrsta        Recall user's status for BLDDSP
14E3A 8F00         GOSBVL =BLDDSP       Rebuild display
      000
14E41 8E00         GOSUBL =SETTMO       Set 10-minute timeout
      00
14E47 8F00  KEYR60 GOSBVL =SLEEP        Wait for key
      000
14E4E 5F2          GONC   KEYR70        Go if key already in buffer
14E51 7481         GOSUB  usrsta        Recall user's status for CKSREQ
14E55 7882         GOSUB  cksreq        Check what woke us.
14E59 8E00         GOSUBL =ALMSRV       Now check if timed out
      00
14E5F 863          ?ST=0  fTMOUT        Timed out?
14E62 5E           GOYES  KEYR60        No, back to sleep.
14E64 3100         LC(2)  =k#OFF        Yes, return ASCII OFF key
14E68 60C1         GOTO   C=K#
14E6C 860   KEYR68 ?ST=0  0             Here if pKYDF was handled.
14E6F 9A           GOYES  KEYR50        Should we return to caller? No.
14E71 01           RTN                  Yes.  Defn pointer in DEFADR.
```

```
405 14E73 110   KEYR69 A=R0                    Here if pWTKY was handled.
406 14E76 D1           B=0    A
407 14E78 AE8          B=A    B
408 14E7B 4C0          GOC    KEYR72           B.E.T.
409 14E7E 8F00 KEYR70 GOSBVL =POPBUF           Pop key from buffer
            000
410 14E85 429          GOC    KEYR50           Go if no key
411 14E88 D4   KEYR72  A=B    A                Key#
412 14E8A 3400         LC(5)  =KEYCOD
            000
413 14E91 C9           C=C+B  A
414 14E93 C9           C=C+B  A                Calculate table entry address
415 14E95 134          D0=C
416 14E98 8E00         GOSUBL =ASLW5           Key# to A[9-5]
            00
417 14E9E 14A          A=DAT0 B                Read keycode from table
418 14EA1 100          R0=A                    Save keycode & key# in R0
419 14EA4 8E00         GOSUBL =FPOLL
            00
420 14EAA 00           CON(2) =pKYDF           Poll before processing key
421 14EAC 831          ?XM=0                   Handled?
422 14EAF DB           GOYES  KEYR68           Yes.
423 14EB1 3100         LC(2)  =flVIEW
424 14EB5 7052         GOSUB  Sflag?           Is VIEW needed?
425 14EB9 460          GOC    VIEW             Yes.
426 14EBC 6980         GOTO   KEYR75           No, process key.
427 14EC0 3100 VIEW    LC(2)  =flVIEW          Yes, then view key
428 14EC4 7742         GOSUB  SFlagC           Turn off VIEW flag
429 14EC8 1F00         D1=(5) (=FUNCR0)-2
            000
430 14ECF D2           C=0    A
431 14ED1 31C2         LC(2)  44
432 14ED5 8E00         GOSUBL =WIPOUT
            00
433 14EDB 74A2         GOSUB  GETDEF
434 14EDF 543          GONC   VIEWUN           Not redefined; show "Unassigned"
435 14EE2 7F22         GOSUB  KEYTYP
436 14EE6 1B00         D0=(5) (=FUNCR0)-2
            000
437 14EED 14C          DAT0=C B                Write out def type indicator
438 14EF0 34B2         LC(5)  21*2+1
            000
439 14EF7 8BA          ?A<=C  A                Def len>21 chars?
440 14EFA 40           GOYES  VIEW20           No, then okay
441 14EFC DA           A=C    A                Yes, then limit it to 21 chars
442 14EFE CC   VIEW20  A=A-1  A                Decrement length count
443 14F00 CC           A=A-1  A
444 14F02 4B3          GOC    VIEW30           At end of text?  Yes, then exit
445 14F05 161          D0=D0+ 2
446 14F08 14F          C=DAT1 B
447 14F0B 171          D1=D1+ 2
448 14F0E 14C          DAT0=C B
449 14F11 5CE          GONC   VIEW20           (B.E.T.)
450 14F14 1B00 VIEWUN  D0=(5) (=FUNCR0)-2
            000
```

```
451 14F1B 3F55          LCASC  \ngissanU\
          E616
          3737
          9676
          E6
452 14F2D 1547          DAT0=C W
453 14F31 16F           D0=D0+ 16
454 14F34 3356          LCASC  \de\
          46
455 14F3A 15C3          DAT0=C 4
456 14F3E 7EF1  VIEW30  GOSUB  VWFC-2        View key
457 14F42 65DE          GOTO   KEYR50
458             *
459 14F46 3100  KEYR75  LC(2)  =flCTRL
460 14F4A 7BB1          GOSUB  Sflag?        Control key hit?
461 14F4E 542           GONC   KEYR80        No.
462 14F51 3100          LC(2)  =flCTRL       Yes.
463 14F55 76B1          GOSUB  SFlagC        Clear it.
464 14F59 110           A=R0                 Fetch keycode
465 14F5C 3304          LCASC  \_@\          In range of CTRL'able keys?
          F5
466 14F62 7D91          GOSUB  range
467 14F66 4C0           GOC    KEYR80        No, process normally.
468 14F69 7A42          GOSUB  FLIP0         Yes, fetch key# to C[B].
469 14F6D 21            P=     1             Indicate this is CTRL key.
470 14F6F 69B0          GOTO   C=K#          CTRL result w/key# in C.
471 14F73 3100  KEYR80  LC(2)  =flUSRX       Temporary USER flag set?
472 14F77 7E81          GOSUB  Sflag?
473 14F7B 561           GONC   KEYR90        No.  Process normally.
474 14F7E 3100          LC(2)  =flUSRX       Yes.
475 14F82 7981          GOSUB  SFlagC        Clear temporary USER.
476 14F86 3100          LC(2)  =flUSER
477 14F8A 79B2          GOSUB  sflagt        Toggle USER.
478 14F8E 6B00          GOTO   KEY100        Carry reflects old state.
479 14F92 3100  KEYR90  LC(2)  =flUSER       USER set?
480 14F96 7F61          GOSUB  Sflag?
481 14F9A 554   KEY100  GONC   KEY110        No.  Process normally.
482 14F9D 72E1          GOSUB  GETDEF        Yes.  Look for definition.
483 14FA1 5E3           GONC   KEY110        Go if no definition.
484 14FA4 171           D1=D1+ 2             Point to length field.
485 14FA7 1533          A=DAT1 X             Read length & keytype.
486 14FAB B24           A=A+1  XS            Convert keytype for KEYRD output.
487 14FAE A24           A=A+A  XS
488 14FB1 D6            C=A    A
489 14FB3 172           D1=D1+ 3             Point to text of key defn.
490 14FB6 137           CD1EX                Addr to C, len to D1[B].
491 14FB9 8E00          GOSUBL =CSLW3        C[7-3]=Addr of text
          00
492 14FBF AA6           C=A    XS            C[2]=keytype (2,4, or 6).
493 14FC2 1C4           D1=D1- 5             Subtract length of record hdr.
494 14FC5 133           AD1EX
495 14FC8 81C           ASRB                 Divide by 2 for bytes
496 14FCB AE6           C=A    B             C[1-0]=definition length.
497 14FCE 1F00  KEYDON  D1=(5) =DEFADR
          000
```

```
498 14FD5 15D7          DAT1=C 8            Write out to RAM
499 14FD9 8D00  usrsta  GOVLNG =USRSTA      Restore user's status
          000
500 14FE0 34E4  KEY110  LC(5)  KEY120
          051
501 14FE7 06            RSTK=C              RTN addr for special processing
502 14FE9 110           A=R0                Fetch keycode
503 14FEC 8E00          GOSUBL =FINDAj
          00
504 14FF2 00            CON(2) =kcLC
505 14FF4 F42           REL(3) LOWERC       Lowercase mode
506 14FF7 00            CON(2) =kcUSER
507 14FF9 452           REL(3) USERK        User mode
508 14FFC 00            CON(2) =kcCTRL
509 14FFE 782           REL(3) CTRL         CTRL key
510 15001 00            CON(2) =kcVIEW
511 15003 C92           REL(3) VIEWK        VIEW key
512 15006 00            CON(2) =kcUSEX
513 15008 582           REL(3) USERX        Temp user mode toggle
514 1500B 00            CON(2) =kcLERR
515 1500D 9E0           REL(3) lerrm        Last error message display
516 15010 00            CON(2) 0
517 15012 07            C=RSTK              No special processing
518 15014 31E7          LCHEX  7E
519 15018 9E6           ?A>C   B            Typing aid?
520 1501B 25            GOYES  NEWTOK       Yes
521 1501D 3100          LC(2)  =flLC        No.  LC mode set?
522 15021 74E0          GOSUB  Sflag?
523 15025 7F61          GOSUB  FLIPCS       Flip case if carry set.
524 15029 D0    C=K#    A=0    A
525 1502B AEA           A=C    B            A[A]=key#
526 1502E 80FF          CPEX   15           Key type to C[S]
527 15032 20            P=     0
528 15034 3400          LC(5)  =KEYCOD
          000
529 1503B C2            C=A+C  A
530 1503D C2            C=A+C  A            Index into keycode table.
531 1503F 812           CSLC
532 15042 BF2           CSL    W
533 15045 BF2           CSL    W            C[7-3]=addr, C[2]=type
534 15048 E6            C=C+1  A            C[1-0]=length = 01
535 1504A 638F          GOTO   KEYDON       Write out to DEFADR.
536 1504E 1B00  KEY120  DO=(5) =KEYPTR      Rtn here from internal key proc.
          000
537 15055 D2            C=0    A            Zero C[B]
538 15057 1562          C=DATO XS           Read keybuffer count
539 1505B 92E           ?C#0   XS           Keybuffer empty?
540 1505E B0            GOYES  KEY130       No.
541 15060 1A00          DO=(4) (=KEYBUF)+2*14
          00
542 15066 14C           DATO=C B            Yes.  Disable repeating.
543 15069 6EAD  KEY130  GOTO   KEYR50
544 1506D 8F00  NEWTOK  GOSBVL =MTADDR
          000
545 15074 D0            A=0    A
```

```
546 15076 1533          A=DAT1 X            Offset from LXTXTT to text
547 1507A 177           D1=D1+ 8            Point at characterization nib
548 1507D 1574          C=DAT1 S
549 15081 08            CLRST
550 15083 85B           ST=1   11           Indicate this is typing aid
551 15086 94E           ?C#0   S            Begin BASIC?
552 15089 50            GOYES  NEWT05       Yes, no space before.
553 1508B 85A           ST=1   10           Indicate leading space needed.
554 1508E 859   NEWT05  ST=1   9            Indicate trailing space needed.
555 15091 B46           C=C+1  S
556 15094 512           GONC   NEWT10       Go if not function
557 15097 849           ST=0   9            Trailing space not needed
558 1509A 1C4           D1=D1- 5
559 1509D 147           C=DAT1 A            Read execution addr offset
560 150A0 133           AD1EX
561 150A3 CA            A=A+C  A
562 150A5 133           AD1EX               Point at execution addr
563 150A8 1C1           D1=D1- 2            Point at parm count
564 150AB 14F           C=DAT1 B
565 150AE 90A           ?C=0   P            Min # parms > 0?
566 150B1 50            GOYES  NEWT10       No
567 150B3 858           ST=1   8            Yes, indicate paren needed
568 150B6 3400  NEWT10  LC(5)  =LXTXTT
          000
569 150BD C2            C=C+A  A            Address of text
570 150BF 135           D1=C
571 150C2 E6            C=C+1  A            Point at actual text
572 150C4 BF2           CSL    W
573 150C7 BF2           CSL    W
574 150CA BF2           CSL    W            Text addr to C[7-3]
575 150CD 09            C=ST                Key type to C[2]
576 150CF D0            A=0    A
577 150D1 15B0          A=DAT1 1            Read textlen-1
578 150D5 E4            A=A+1  A
579 150D7 81C           ASRB                Textlen in bytes
580 150DA A86           C=A    P            C[1-0]=text length
581 150DD 60FE          GOTO   KEYDON
582             *
583 150E1 8E00  cksreq  GOSUBL =savlvl
          00
584 150E7 8F00          GOSBVL =CKSREQ
          000
585 150EE 8E00          GOSUBL =rstlvl
          00
586 150F4 01            RTN                 This cannot be packed out!!!
587             ********************************************************************
588             ********************************************************************
589             **
590             ** Name:(S) pWTKY   -  Poll When Waiting For Key
591             **
592             ** Category:   POLL
593             **
594             ** Type:       FPOLL
595             **
596             ** Purpose:
```

```
597             **      Allow LEXFILE to circumvent waiting for and fetching
598             **      ket# in KEYRD.
599             **
600             ** Should poll be "Handled" (return with XM=0)?:
601             **      Yes, if LEXFILE wishes to "press" a key.
602             **
603             ** Meaning of "Handling" Poll (what does code do if handled?):
604             **      Lexfile is "press"ing a key.  If poll is handled,
605             **      KEYRD goes on to process key returned by this poll.
606             **
607             **      If poll is not handled, KEYRD will look for repeating
608             **      keys.  Seeing none, KEYRD will pop the next key# from
609             **      the keybuffer or, if buffer is empty, wait until a key
610             **      is hit and then process it.
611             **
612             ** Entry conditions for handler (registers, ST, RAM, etc.):
613             **      Carry set.
614             **      B[A] = Poll number.
615             **      HEX mode.
616             **      P=0.
617             **
618             ** Normal exit conditions from handler if handled (ST, RAM,
619             ** registers, etc.):
620             **      HEX mode.
621             **      XM=0.
622             **      R0[B] contains key# (physical keycode).
623             **
624             ** Normal exit conditions from handler if not handled (ST, RAM,
625             ** registers, etc.):
626             **      HEX mode.
627             **      XM=1.
628             **
629             ** Available subroutine levels:
630             **      2
631             **
632             ** NOTE:
633             **      We are just entering KEYRD when this poll occurs.
634             **
635             **      This is the time to press a key.  The time to provide
636             **      a definition for a pressed key is the pKYDF poll.
637             **
638             ** What registers/RAM may be used if handled?:
639             **      A-D, D0, D1, P, R0, R3.
640             **      SCRTCH RAM.
641             **
642             ** What registers/RAM may be used if not handled?:
643             **      A-C, D[15-5] D0, D1, P, R0, R3.
644             **      SCRTCH RAM.
645             **
646             ** Special memory/pointer considerations (are pointers funny?):
647             **      May be in CALC mode.
648             **
649             ** Envisioned application(s):
650             **      External keyboard controller or remote keyboard.
651             **      The poll handler may take over waiting for a key to
```

```
652             **        go down if appropriate.
653             **
654             ** History:
655             **
656             **    Date     Programmer              Modification
657             **  --------   ----------   ----------------------------------
658             **  05/19/83   NM           Added documentation
659             **
660             ********************************************************************
661             ********************************************************************
662             ********************************************************************
663             ********************************************************************
664             **
665             ** Name:(S) pKYDF   -  Poll To Define Key
666             **
667             ** Category:   POLL
668             **
669             ** Type:       FPOLL
670             **
671             ** Purpose:
672             **      Allow LEXFILE to define action/definition of a key.
673             **
674             ** Should poll be "Handled" (return with XM=0)?:
675             **      Yes, if you want to define or act on the key.
676             **
677             ** Meaning of "Handling" Poll (what does code do if handled?):
678             **      LEXFILE is either defining or otherwise acting on key.
679             **      Defining (returning with SO=1) means that the LEXFILE
680             **        is returning a definition to whomever called KEYRD
681             **        (CHEDIT, CALC mode editor, or whoever).
682             **      Acting on (returning with SO=0) means that the LEXFILE
683             **        is using the key in some way (such as toggling a
684             **        flag or ignoring) and KEYRD should not return a
685             **        definition to the caller, but should instead get the
686             **        next key to process.
687             **
688             ** Entry conditions for handler (registers, ST, RAM, etc.):
689             **      Carry set.
690             **      B[A] = Poll number.
691             **      HEX mode.
692             **      P=0.
693             **      R0[A]=keycode (from keycode map),
694             **      R0[9-5]=key# (physical keycode).
695             **
696             ** Normal exit conditions from handler if handled (ST, RAM,
697             ** registers, etc.):
698             **      HEX mode.
699             **      XM=0.
700             **      If handled but not returning a definition ("acting on"
701             **        a key): SO=0.
702             **      If returning a definition:
703             **        SO=1.
704             **        Definition pointer in DEFADR (in RAM) as follows:
705             **          DEFADR: Length of string in bytes (2 nibs).
706             **          DEFADR+2: Key type (1 nib).
```

```
707             **                      0 = Single ASCII character.  Includes
708             **                          control chars 0-31, which may cause
709             **                          some action by caller.
710             **                      1 = ASCII control char + #40.  This is
711             **                          a character in the range 0-31 which
712             **                          is to be interpreted strictly as a
713             **                          character, not as special action keys
714             **                          (cursor-right, etc.).  To return
715             **                          char #01, DEFADR should point at #41
716             **                          byte, etc.
717             **                      2 = User defined key--terminating.
718             **                      4 = User defined key--non-terminating.
719             **                      6 = User defined key--immed execute.
720             **                      8-F = LEX table entry, with lower 3 bits
721             **                            as follows:
722             **                              0: Parentheses needed.
723             **                              1: Trailing space needed.
724             **                              2: Leading space needed.
725             **              DEFADR+3: Address of text.
726             **
727             ** Normal exit conditions from handler if not handled (ST, RAM,
728             ** registers, etc.):
729             **      HEX mode.
730             **      XM=1.
731             **
732             ** Available subroutine levels:
733             **      2
734             **
735             ** What registers/RAM may be used if handled?:
736             **      A-D, D0, D1, P, R0, R3.
737             **      SCRTCH RAM.
738             **
739             ** What registers/RAM may be used if not handled?:
740             **      A-C, D[15-5] D0, D1, P, R3.
741             **      SCRTCH RAM.
742             **
743             ** Special memory/pointer considerations (are pointers funny?):
744             **      May be in CALC mode.
745             **
746             ** Envisioned application(s):
747             **      Redefine keyboard.
748             **
749             **      One interesting application: Stuff funny key# in
750             **      keybuffer (perhaps at pSREQ) and define it here.
751             **
752             ** History:
753             **
754             **    Date     Programmer             Modification
755             **  --------   ----------   ------------------------------------
756             **  05/19/83   NM           Added documentation
757             **
758             ****************************************************************
759             ****************************************************************
760             *
761 150F6 8D00  lerrm  GOVLNG =LERRM
```

```
            000
    762 150FD 3316 =aRANGE LCASC  \za\
            A7
    763 15103 8C00  range  GOLONG =RANGE
            00
    764 15109 8C00 =Sflag? GOLONG =SFLAG?
            00
    765 1510F 8C00 =SFlagC GOLONG =SFLAGC
            00
    766           *
```

```
                    STITLE KEYTYP
                ****************************************************************
                ****************************************************************
                **
                ** Name:    KEYTYP  -  Key Definition Type (ASCII Form)
                **
                ** Category:   KEYUTL
                **
                ** Purpose:
                **      Returns an ASCII letter indicating key definition type
                **
                ** Entry:
                **      HEX mode.
                **      C(A)=Entire definition length
                **      D1 points at definition entry
                **
                ** Exit:
                **      P = 0
                **      Carry set
                **      C(B)=\ \ if terminating key definition
                **           \;\ if non-terminating key definition
                **           \:\ if immediate execute key definition
                **      A(S)=Key definition type (0=term,1=non-term,2=im. ex)
                **      A(A)=Actual key definition length+1 (nibs)
                **      D1 points at key definition.
                **
                ** Calls:      None
                **
                ** Uses.......
                **              A,C,D1,P
                **
                ** Stk lvls:   0
                **
                ** Detail:
                **      This subroutine should be called following a KEYFND.
                **      It does 3 good things:
                **           1 -- Finds ASCII definition type
                **           2 -- Moves D1 to point at actual definition
                **           3 -- Subtracts 4 from definition length which
                **                leaves it one greater than the number of
                **                nibbles in the actual definition.  This
                **                number can then be double-decremented until
                **                end of definition and will carry on second
                **                decrement of pair.
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **   --------   ----------   ----------------------------------
                **   07/06/82   BS           Added documentation
                **   11/03/82   NM           Added magic LC
                **
                ****************************************************************
                ****************************************************************
 15115 CE       =KEYTYP C=C-1  A
```

```
    822 15117 CE           C=C-1  A
    823 15119 CE           C=C-1  A
    824 1511B CE           C=C-1  A              Get actual string length+1
    825 1511D 173          D1=D1+ 4
    826 15120 1574         C=DAT1 S              C[S]=key definition type
    827 15124 AFA          A=C    W              Hold in A
    828 15127 170          D1=D1+ 1              Increment past def type
    829 1512A A46  =KYTYP1 C=C+C  S
    830 1512D BCA          C=-C   S              0->0, 1->E, 2->C
    831 15130 80DF         P=C    15
    832 15134 3502         LCASC  \:; \
              B3A3
    833 1513C 20           P=     0
    834 1513E 02           RTNSC
```

```
                        STITLE View a buffer
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) VIEWD1  -  View A Buffer While Keys Down
                **
                ** Category:   DSPUTL
                **
                ** Purpose:
                **      This entry point takes a 22 character buffer pointed to
                **      by D1 and builds a bit pattern in display inside
                **      the WINDOW setting.  This display is held until all
                **      keys are up.
                **
                ** Entry:
                **      P = 0
                **      D1 points at a 22 character buffer.
                **
                ** Exit:
                **      P      =  0
                **
                ** Calls:      BLDBIT
                **
                ** Uses.......
                **              A,B,C,D,D0,D1
                **
                ** Stk lvls:   2
                **
                ** Detail:
                **      This routine looks at the current WINDOW settings
                **      to set up the first character position and the number
                **      of characters to be displayed.  Since this may be
                **      (and usually is) 22 characters, the buffer to be viewed
                **      should be at least 22 characters.  It should be padded
                **      with either blanks or nulls to prevent unwanted "junk"
                **      at the end of the display.
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **   --------   ----------   ----------------------------------
                **   07/15/82   BS           Updated documentation
                **
                ********************************************************************
                ********************************************************************
15140 1F00  VWFC-2 D1=(5) (=FUNCR0)-2
      000
15147 1B00 =VIEWD1 D0=(5) =WINDLN       Prepare to build bit pattern
      000
1514E 14E          C=DAT0 B
15151 816          CSRC
15154 816          CSRC                 Put length in C(14,15)
15157 1900         D0=(2) =WINDST
1515B D2           C=0    A
1515D 14E          C=DAT0 B             Read starting position
```

```
    888 15160 AF7          D=C    W              Set up D for BLDBIT
    889 15163 137          CD1EX                 C(A)=Start of buffer
    890 15166 8F00         GOSBVL =BLDBIT
              000
    891 15170 8F00         GOSBVL =RCLSTA
              000
    892 15174 841          ST=0   BitsOK         Display is not exactly right
    893 15177 0B           CSTEX
    894 15179 1543         DAT0=C X
    895 1517D 8C00 =waitky GOLONG =WAITKY        Wait for all keys up
              00
```

```
                        STITLE GETDEF
                ********************************************************************
                ********************************************************************
                **
                ** Name:    GETDEF  -  Check Upper/Lower Case, Get Key Def
                **
                ** Category:   KEYUTL
                **
                ** Purpose:
                **      Checks for keys being case toggled before looking up
                **      key definition.
                **
                ** Entry:
                **      P = 0
                **      R0 = Key code
                **      R3 = Key number
                **
                ** Exit:
                **      P = 0
                **      B(A) = Keycode
                **      Carry clear ==> Key not redefined
                **            set   ==> Key redefinition found
                **                      D1 points to defn record in keyfile
                **                      C(A)=Entire entry length
                **                      D0 points to file header end
                **
                ** Calls:      SFLAG?,CONVLC,CONVUC,KEYFND
                **
                ** Uses.......
                **              A,B,C,D,D0,D1,S6,S8,R3
                **
                ** Stk lvls:   2
                **
                ** Algorithm:
                **      If flLC(ie Lower case mode) then
                **          If Character is upper case letter
                **              then add 56*2 to key number (convert to LC)
                **          else if character is lower case letter
                **              then subtract 56*2 from key number (convert to
                **              UC).
                **      Call KEYFND to find key definition(if any)
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **   --------   ----------   ----------------------------------
                **  07/15/82    BS           Added documentation
                **
                ********************************************************************
                ********************************************************************
15183 3100  GETDEF LC(2)  =flLC
15187 7E7F          GOSUB  Sflag?          Lower case mode?
1518B 7900          GOSUB  FLIPCS          Flip case if carry set
1518F D5            B=C    A               Read converted key#
15191 8D00 =keyfnd GOVLNG =KEYFND
```

```
            000
951              *
952 15198 5E1   FLIPCS GONC   FLIPO          Not in lowercase mode.
953 1519B 3107         LC(2)  112
954 1519F D5           B=C    A
955 151A1 110          A=R0                  Get keycode
956 151A4 8F00         GOSBVL =ARANGE        Uppercase?
            000
957 151AB 5D0          GONC   FLIPCX         Yes, will add 112 to key#
958 151AE F9           B=-B   A
959 151B0 794F         GOSUB  aRANGE         Lowercase?
960 151B4 540          GONC   FLIPCX         Yes, will add -112 to key#
961 151B7 D1    FLIP0  B=0    A              No. Will add 0.
962 151B9 118   FLIPCX C=R0
963 151BC 8E00         GOSUBL =CSRW5         Retrieve key#
            00
964 151C2 A69          C=B+C  B              Flip case as necessary.
965 151C5 01           RTN
```

```
966                     STITLE Cursor right and left
967 151C7 3134 =CURSRR  LCASC  \C\
968 151CB 62R0          GOTO   escseq
969                *
970 151CF 3144  CURSRL  LCASC  \D\
971 151D3 6R90          GOTO   escseq
```

```
                        STITLE Cursor far right
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) CURSFR  -  Move Cursor To Far Right
                **
                ** Category:   DSPUTL
                **
                ** Purpose:
                **      Send CURSOR FAR RIGHT to display.
                **
                ** Entry:
                **      P = 0
                **      HEX mode.
                **
                ** Exit:
                **      P = 0
                **      Carry clear
                **
                ** Calls:      ESCSEQ (falls through)
                **
                ** Uses.......
                **              A,B,C,D,D0,D1.
                **
                ** Stk lvls:   4
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **   --------   ----------   ---------------------------------
                **   07/15/82   BS           Added documentation
                **
                ********************************************************************
                ********************************************************************
151D7 3130      =CURSFR LCHEX  03
151DB 6290              GOTO   escseq
```

```
1008                    STITLE Cursor far left
1009                ********************************************************************
1010                ********************************************************************
1011                **
1012                ** Name:(S) CURSFL  -  Move Cursor To Far Left
1013                **
1014                ** Category:   DSPUTL
1015                **
1016                ** Purpose:
1017                **      Send CURSOR FAR LEFT to display.
1018                **
1019                ** Entry:
1020                **      P = 0
1021                **
1022                ** Exit:
1023                **      P = 0
1024                **      Carry clear
1025                **
1026                ** Calls:      ESCSEQ (falls through)
1027                **
1028                ** Uses.......
1029                **              A,B,C,D,D0,D1.
1030                **
1031                ** Stk lvls:   4
1032                **
1033                ** History:
1034                **
1035                **    Date      Programmer              Modification
1036                **   --------   ----------   ----------------------------------
1037                **  07/15/82    BS           Added documentation
1038                **  11/04/82    NM           Packed a little.
1039                **  12/09/82    NM           Packed a lot.
1040                **
1041                ********************************************************************
1042                ********************************************************************
1043 151DF 3140 =CURSFL LCHEX  04
1044 151E3 6A80         GOTO   escseq
```

```
1045                    STITLE Backspace
1046                ********************************************************************
1047                ********************************************************************
1048                **
1049                ** Name:    BCKSPC  -  Implement Backspace
1050                **
1051                ** Category:   LOCAL
1052                **
1053                ** Purpose:
1054                **      Implement backspace in CHEDIT.
1055                **
1056                ** Entry:
1057                **      P=0.
1058                **
1059                ** Exit:
1060                **      P=0.
1061                **
1062                ** Calls:      DSPCHC, RCLSTA, DO=CUR, BF2DSP (falls through)
1063                **
1064                ** Uses.......
1065                **              A,B,C,D,D0,D1.
1066                **
1067                ** Stk lvls:   3
1068                **
1069                ** Detail:
1070                **      If insert mode, backspace = control-h, delete char.
1071                **      If replace mode and cursor is at null,
1072                **        backspace = control-h, delete char
1073                **      Else
1074                **        backspace = control-h, space, control-h.
1075                **
1076                ** History:
1077                **
1078                **    Date     Programmer              Modification
1079                **  --------   ----------   ----------------------------------
1080                **  11/04/82   NM           Rewrote slightly
1081                **
1082                ********************************************************************
1083                ********************************************************************
1084 151E7 1F00  BCKSPC D1=(5) =CURSOR
           000
1085 151EE 14F          C=DAT1 B
1086 151F1 108          R0=C
1087 151F4 3180         LC(2)  8                Backspace
1088 151F8 8F00         GOSBVL =DSPCHC          Send backspace to display
           000
1089 151FF 1F00         D1=(5) =CURSOR
           000
1090 15206 14B          A=DAT1 B                Re-read (CURSOR)
1091 15209 118          C=R0                    Recall last value
1092 1520C 962          ?A=C   B                Did backspace move cursor?
1093 1520F 00           RTNYES                  No, then return
1094 15211 8F00         GOSBVL =RCLSTA          Read display status
           000
1095 15218 877          ?ST=1  Insert           Are we in insert mode?
```

```
1096 1521B 26          GOYES  -CHAR          Yes, do delete char.
1097 1521D 8F00        GOSBVL =DO=CUR        No.
           000
1098 15224 161         DO=DO+ 2              Point to char after cursor.
1099 15227 14E         C=DATO B
1100 1522A 96A         ?C=0   B              Null?
1101 1522D 05          GOYES  -CHAR          Yes, do -char.
1102 1522F 1FD3        D1=(5) BCKSTR         Replace mode backspace string
           251
1103 15236 8D00 =bf2dsp GOVLNG =BF2DSP       Send string to display
           000
1104            *
1105 1523D 0280  BCKSTR NIBHEX 0280FF        Space, Backspace
           FF
```

```
1106                    STITLE Lowercase toggle
1107                ********************************************************************
1108                ********************************************************************
1109                **
1110                ** Name:    LOWERC  -  Toggle Lower Case Mode
1111                **
1112                ** Category:   KEYUTL
1113                **
1114                ** Purpose:
1115                **      Toggles lower case mode flag and annunciator.
1116                **
1117                ** Entry:
1118                **      P       =  0
1119                **
1120                ** Exit:
1121                **      P       =  0
1122                **
1123                ** Calls:      SFLAGT (falls through)
1124                **
1125                ** Uses.......
1126                **              A(A),B(A),C(W),D(A)
1127                **
1128                ** Stk lvls:   2
1129                **
1130                ** History:
1131                **
1132                **    Date      Programmer              Modification
1133                **  --------    ----------   ----------------------------------
1134                **  07/16/82    BS           Updated documentation
1135                **
1136                ********************************************************************
1137                ********************************************************************
1138 15243 3100 =LOWERC LC(2)  =flLC
1139 15247 8C00  sflagt GOLONG =SFLAGT
           00
```

```
1140                    STITLE User mode toggle
1141                ********************************************************************
1142                ********************************************************************
1143                **
1144                ** Name:    USERK   -  User Mode Toggle
1145                **
1146                ** Category:   KEYUTL
1147                **
1148                ** Purpose:
1149                **      Toggles user mode flag and annunciator
1150                **
1151                ** Entry:
1152                **      HEX mode.
1153                **      P       =  0
1154                **
1155                ** Exit:
1156                **      P       =  0
1157                **
1158                ** Calls:       SFLAGT
1159                **
1160                ** Uses.......
1161                **              A(A),B(A),C(W),D(A)
1162                **
1163                ** Stk lvls:   2
1164                **
1165                **    Date      Programmer              Modification
1166                **  --------    ----------   ------------------------------------
1167                **  07/16/82    BS           Updated documentation
1168                **
1169                ********************************************************************
1170                ********************************************************************
1171 1524D 3100 =USERK  LC(2)  =flUSER
1172 15251 65FF        GOTO   sflagt
```

```
1173                    $TITLE Insert/Replace toggle
1174                ****************************************************************
1175                ****************************************************************
1176                **
1177                ** Name:    IN/REP  -  Insert/Replace Toggle
1178                **
1179                ** Category:   KEYUTL
1180                **
1181                ** Purpose:
1182                **      Toggles insert/replace mode by sending appropriate
1183                **      escape sequence to display.
1184                **
1185                ** Entry:
1186                **      P       =  0
1187                **
1188                ** Exit:
1189                **      P       =  0
1190                **
1191                ** Calls:      ESCSEQ
1192                **
1193                ** Uses.......
1194                **  Exclusive: A(A),C(A),D0
1195                **  Inclusive: A(W),B(W),C(W),D(W),D0,D1
1196                **
1197                ** Stk lvls:   4
1198                **
1199                ** Detail:
1200                **      Reads display status and sends an ESQ R to display
1201                **      if display was in insert mode and an ESQ Q if display
1202                **      was in replace mode.
1203                **
1204                ** History:
1205                **
1206                **    Date      Programmer              Modification
1207                **  --------    ----------   ----------------------------------
1208                **  07/16/82    BS           Added documentation
1209                **  06/02/83    BS           Rewrote for Mtn. Comp.
1210                **
1211                ****************************************************************
1212                ****************************************************************
1213 15255 8F00 =IN/REP GOSBVL =RCLSTA
           000
1214 1525C 867          ?ST=0  Insert        Set carry if not in insert mode
1215 1525F 20           GOYES  I/R10
1216 15261 0A    I/R10  ST=C
1217 15263 31E4         LCASC  \N\           Escape N = Insert mode (w/ wrap)
1218 15267 460          GOC    I/R20
1219 1526A 3125         LCASC  \R\           Escape R = Replace mode
1220 1526E       I/R20
1221 1526E 8D00  escseq GOVLNG =ESCSEQ
           000
```

```
1222                    STITLE Delete through EOL
1223                ********************************************************************
1224                ********************************************************************
1225                **
1226                ** Name:(S) -LINE   -  Delete Through End Of Line
1227                **
1228                ** Category:   KEYUTL
1229                **
1230                ** Purpose:
1231                **      Send an ESC K to display to delete through end of line
1232                **
1233                ** Entry:
1234                **      P       = 0
1235                **
1236                ** Exit:
1237                **      P       = 0
1238                **
1239                ** Calls:      ESCSEQ
1240                **
1241                ** Uses.......
1242                **  Exclusive: C(B)
1243                **  Inclusive: A(W),B(W),C(W),D(W),D0,D1
1244                **
1245                ** Stk lvls:   4
1246                **
1247                ** History:
1248                **
1249                **    Date      Programmer              Modification
1250                **  --------    ----------   ----------------------------------
1251                **  07/16/82    BS           Added documentation
1252                **
1253                ********************************************************************
1254                ********************************************************************
1255 15275 31B4 =-LINE  LCASC  \K\             Escape K = Delete through eol
1256 15279 64FF        GOTO   escseq
```

```
1257                    STITLE Delete Character
1258                ********************************************************************
1259                ********************************************************************
1260                **
1261                ** Name:     -CHAR   -  Delete Char Key
1262                **
1263                ** Category:   KEYUTL
1264                **
1265                ** Purpose:
1266                **       Sends an ESC P to display to delete a character.
1267                **       Builds display to reflect modified buffer.
1268                **
1269                ** Entry:
1270                **       P      = 0
1271                **
1272                ** Exit:
1273                **       P      = 0
1274                **
1275                ** Calls:      ESCSEQ
1276                **
1277                ** Uses.......
1278                **  Exclusive: C(B)
1279                **  Inclusive: R3,A,B,C,D,D0,D1
1280                **
1281                ** Stk lvls:   4
1282                **
1283                ** History:
1284                **
1285                **    Date     Programmer             Modification
1286                **  --------   ----------   ----------------------------------
1287                **  07/16/82   BS           Added documentation
1288                **
1289                ********************************************************************
1290                ********************************************************************
1291 1527D 31F4 =-CHAR  LCASC  \O\            Escape O = Delete char (w/ wrap)
1292 15281 6CEF        GOTO   escseq
```

```
1293                    STITLE CTRL flag set
1294                ********************************************************************
1295                ********************************************************************
1296                **
1297                ** Name:    CTRL    -  Sets Control Flag
1298                **
1299                ** Category:   KEYUTL
1300                **
1301                ** Purpose:
1302                **      Sets control flag (flCTRL) so next key will be
1303                **      CTRL'd.
1304                **
1305                ** Entry:
1306                **      HEX mode.
1307                **      P       =  0
1308                **
1309                ** Exit:
1310                **      P       =  0
1311                **
1312                ** Calls:      SFLAGS
1313                **
1314                ** Uses.......
1315                **              A(A),B(A),C(W),D(A)
1316                **
1317                ** Stk lvls:   2
1318                **
1319                ** History:
1320                **
1321                **    Date      Programmer              Modification
1322                **   --------   ----------   ----------------------------------
1323                **  07/16/82    BS           Added documentation
1324                **
1325                ********************************************************************
1326                ********************************************************************
1327 15285 3100 =CTRL   LC(2)  =flCTRL         Set control flag
1328 15289 6F00         GOTO   SFlagS
```

```
1329                    STITLE Single-key user mode set
1330                ****************************************************************
1331                ****************************************************************
1332                **
1333                ** Name:    USERX   -  Set Single Key User Mode
1334                **
1335                ** Category:   KEYUTL
1336                **
1337                ** Purpose:
1338                **      Sets single key user mode by setting single key user
1339                **      flag and toggling USER flag and annunciator.
1340                **
1341                ** Entry:
1342                **      P       =  0
1343                **
1344                ** Exit:
1345                **      P       =  0
1346                **
1347                ** Calls:       SFLAGS,SFLAGT
1348                **
1349                ** Uses.......
1350                **              A(A),B(A),C(W),D(A)
1351                **
1352                ** Stk lvls:   4
1353                **
1354                ** History:
1355                **
1356                **    Date      Programmer              Modification
1357                **  --------    ----------   ----------------------------------
1358                **  07/16/82    BS           Added documentation
1359                **
1360                ****************************************************************
1361                ****************************************************************
1362 15280 3100 =USERX  LC(2)  =flUSER
1363 15291 72BF         GOSUB  sflagt          Toggle USER mode flag
1364 15295 3100         LC(2)  =flUSRX
1365 15299 8C00  SFlagS GOLONG =SFLAGS         Set single key user flag
           00
```

```
                        STITLE View flag set
                ********************************************************************
                ********************************************************************
                **
                ** Name:    VIEWK   -  View Key
                **
                ** Category:   KEYUTL
                **
                ** Purpose:
                **      Sets view flag to enable viewing next keystroke
                **
                ** Entry:
                **      P       =  0
                **      HEX mode.
                **
                ** Exit:
                **      P       =  0
                **
                ** Calls:      SFLAGS (falls through)
                **
                ** Uses.......
                **              A(A),B(A),C(W),D(A)
                **
                ** Stk lvls:   2
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **   --------   ----------   ---------------------------------
                **   07/16/82   BS           Added documentation
                **
                ********************************************************************
                ********************************************************************
1529F 3100 =VIEWK  LC(2)  =flVIEW
152A3 65FF         GOTO   SFlagS         Set VIEW flag
```

```
1401                    STITLE Convert to uppercase
1402                ****************************************************************
1403                ****************************************************************
1404                **
1405                ** Name:(S) CONVUC  -  Convert To Upper Case
1406                ** Name:(S) CNVUCR  -  Convert To Upper Case
1407                **
1408                ** Category:   CONVRT
1409                **
1410                ** Purpose:
1411                **      Convert char in A(B) to upper case if lower case
1412                **      Read a byte into a first if CNVUCR entry point used.
1413                **
1414                ** Entry:
1415                **      A(B) = Character to be converted
1416                **      P    = 0
1417                **      HEX mode.
1418                **
1419                ** Exit:
1420                **      Carry set if no conversion required
1421                **      A(B)=converted letter, not changed if carry set at
1422                **        exit.
1423                **      P    = 0
1424                **
1425                ** Calls:      Range
1426                **
1427                ** Uses.......
1428                **  Exclusive: C(3-0),A(B)
1429                **  Inclusive: C(A),A(B)
1430                **
1431                ** Stk lvls:   1
1432                **
1433                ** History:
1434                **
1435                **    Date     Programmer             Modification
1436                **  --------   ----------   ---------------------------------
1437                **  07/16/82   BS           Updated documentation
1438                **
1439                ****************************************************************
1440                ****************************************************************
1441 152A7 14B   =CNVUCR A=DAT1 B              Read a byte
1442 152AA 7F4E  =CONVUC GOSUB  aRANGE         Lowercase?
1443 152AE 400          RTNC                   No.
1444 152B1 3102         LCHEX  20
1445 152B5 B6A          A=A-C  B
1446 152B8 03           RTNCC
```

```
1447                    STITLE RPTKY
1448                ********************************************************************
1449                ********************************************************************
1450                **
1451                ** Name:(S) RPTKY   -  Check For Repeating Keys
1452                **
1453                ** Category:   KEYUTL
1454                **
1455                ** Purpose:
1456                **      Check for repeating keys.
1457                **
1458                ** Entry:
1459                **      P=0.
1460                **      HEX mode.
1461                **      The last position of the keybuffer contains the key#
1462                **        to look for.
1463                **      System flag flRPTD indicates whether the key has begun
1464                **        repeating yet.
1465                **      User status bits have been saved into DSPSTA.
1466                **
1467                ** Exit:
1468                **      Carry clear if: Key comes up before repeat interval.
1469                **                      Keybuffer non-empty.
1470                **                      No key in last position of keybuffer.
1471                **      Carry set indicates that a repeat should be done.
1472                **        Key# is in B[A].
1473                **      Flag flRPTD = 1 iff carry set.
1474                **      P=0.
1475                **      TIMER1 has been reset to .5 sec.
1476                **      User status bits have NOT been restored to ST.
1477                **
1478                ** Calls:      CKSREQ, DEBNCE, IDIVA, TMRRST, WRTTM1, Sflag?,
1479                **             SFlagC, SFlagS, usrsta.
1480                **
1481                ** Uses.......
1482                **             A,B,C,D,P,D0,D1,ST
1483                **
1484                ** Stk lvls:   3
1485                **
1486                ** History:
1487                **
1488                **    Date     Programmer              Modification
1489                **  --------   ----------   ---------------------------------
1490                **  11/04/82   NM           Wrote.
1491                **
1492                ********************************************************************
1493                ********************************************************************
1494 152BA          =RPTKY
1495 152BA 3100            LC(2)  =flRPTD
1496 152BE 774E            GOSUB  Sflag?          Has key already repeated?
1497 152C2 3572            LCHEX  700027          Short repeat interval.
           0007
1498 152CA 460             GOC    RPTK10          Go if yes.
1499 152CD 31FD            LCHEX  DF              Long repeat interval.
1500 152D1 76B0  RPTK10    GOSUB  WRTTM1          Load timer w/rpt interval
```

```
1501 152D5 3100          LC(2)  =flRPTD
1502 152D9 7CBF          GOSUB  SFlagS          Indicate key has repeated
1503 152DD 28    RPTK20  P=     8              For debounce time
1504 152DF 8F00          GOSBVL =DEBNCE         Scan keyboard
           000
1505 152E6 1F00          D1=(5) (=KEYBUF)+2*14
           000
1506 152ED D0            A=0    A
1507 152EF 14B           A=DAT1 B              Read key#
1508 152F2 CC            A=A-1  A              Key#-1
1509 152F4 D2            C=0    A
1510 152F6 31E0          LC(2)  14
1511 152FA 8E00          GOSUBL =IDIVA         Col# to B, row# to A
           00
1512 15300 20            P=     0
1513 15302 3400          LC(5)  =KCOLO
           000
1514 15309 E9            C=C-B  A
1515 1530B 135           D1=C                  Point at desired column
1516 1530E 1572          C=DAT1 XS             Read column nibble
1517 15312 303           LCHEX  3
1518 15315 0E06          A=A&C  P              Row# mod 4 (unshifts key)
1519 15319 A0C   RPTK30  A=A-1  P              Loop to shift row bit to C:11
1520 1531C 490           GOC    RPTK40
1521 1531F A26           C=C+C  XS
1522 15322 66FF          GOTO   RPTK30
1523 15326 A26   RPTK40  C=C+C  XS             Shift off row bit
1524 15329 441           GOC    RPTK45         Go if key is still down
1525 1532C 3100  RPTKCC  LC(2)  =flRPTD        Here to RTNCC
1526 15330 7BDD          GOSUB  SFlagC          No more repeat
1527 15334 AF2   TMRRST  C=0    W              Here to reset timer
1528 15337 B26           C=C+1  XS             C=000100
1529 1533A 6050          GOTO   WRTTM1         Set .5 sec delay
1530 1533E 1B00  RPTK45  DO=(5) (=TIMER1)+5
           000
1531 15345 1F00          D1=(5) =KEYPTR
           000
1532 1534C 1520          A=DATO P              Timer hinib.
1533 15350 307           LCHEX  7
1534 15353 902           ?A=C   P              Timed out?
1535 15356 02            GOYES  RPTK47         No.
1536 15358 78DF          GOSUB  TMRRST         Set timer to .5 sec
1537 1535C 1B00          DO=(5) (=KEYBUF)+2*14
           000
1538 15363 1532          A=DAT1 XS
1539 15367 92C           ?A#0   XS             Buffer still empty?
1540 1536A 2C            GOYES  RPTKCC         No.
1541 1536C 14A           A=DATO B              Yes.  This will get right key.
1542 1536F D1            B=0    A
1543 15371 AE8           B=A    B              Key# to B[A]
1544 15374 02            RTNSC
1545 15376 1532  RPTK47  A=DAT1 XS             Look at keybuffer.
1546 1537A 92C           ?A#0   XS             Empty?
1547 1537D FA            GOYES  RPTKCC         No.  RTNCC.
1548 1537F 765C          GOSUB  usrsta         Recall user's status for CKSREQ
```

```
1549 15383 7A5D         GOSUB  cksreq         Check SREQs and go again.
1550 15387 655F         GOTO   RPTK20
1551              ****************************************************************
1552              ****************************************************************
1553              **
1554              ** Name:     WRTTMR  -  Write A Value To A Timer
1555              ** Name:     WRTTM1  -  Write A Value To A Timer
1556              **
1557              ** Category:   GENUTL
1558              **
1559              ** Purpose:
1560              **      Write a value to a display driver timer.  Handles
1561              **      possibility that timer kerchunks.  WRTTMR writes to
1562              **      any timer, WRTTM1 writes to TIMER1.
1563              **
1564              ** Entry:
1565              **   WRTTMR: D0 points at timer.
1566              **   BOTH:   Value to write in C[0-5].
1567              **
1568              ** Exit:
1569              **      P=0.
1570              **      Carry clear.
1571              **
1572              ** Calls:      None.
1573              **
1574              ** Uses.......
1575              **              A[0-5], D0 (WRTTM1 only).
1576              **
1577              ** Stk lvls:   0
1578              **
1579              ** History:
1580              **
1581              **    Date      Programmer              Modification
1582              **  --------    ----------   ------------------------------------
1583              **  01/25/83    NM           Created.
1584              **
1585              ****************************************************************
1586              ****************************************************************
1587 1538B 1B00 =WRTTM1 D0=(5) =TIMER1
          000
1588 15392 25   =WRTTMR P=     5
1589 15394 15C5         DAT0=C 6              Write out timer value.
1590 15398 15A5         A=DAT0 6              Read back.
1591 1539C 912          ?A=C   WP             Match?
1592 1539F 60           GOYES  WRTTMX         Yes, done.
1593 153A1 15C5         DAT0=C 6              No.  Write again.
1594 153A5 20    WRTTMX P=     0              Done.
1595 153A7 03           RTNCC
1596 153A9              END
```

```
=-CHAR   Abs   86653 #1527D -  1291   196  1096  1101
=-LINE   Abs   86645 #15275 -  1255   200
 ALMSRV  Ext                -   397
 ARANGE  Ext                -   956
 ASLW5   Ext                -   416
 BCKSPC  Abs   86503 #151E7 -  1084   203
 BCKSTR  Abs   86589 #1523D -  1105  1102
 BF2DSP  Ext                -  1103
 BLDBIT  Ext                -   890
 BLDDSP  Ext                -   379   391
 BitsOK  Abs       1 #00001 -    13   892
 C=K#    Abs   86057 #15029 -   524   401   470
 CHED00  Abs   85163 #14CAB -   133   152   155   187   192   211
 CHED10  Abs   85209 #14CD9 -   147   145
 CHED30  Abs   85222 #14CE6 -   151   149
 CHED40  Abs   85235 #14CF3 -   155   164
 CHED50  Abs   85239 #14CF7 -   156   143
 CHED60  Abs   85279 #14D1F -   170   157
 CHED70  Abs   85329 #14D51 -   188   171
 CHED80  Abs   85389 #14D8D -   207   191
 CHEDEX  Abs   85130 #14C8A -    57   168   182
 CHEDEj  Abs   85273 #14D19 -   168   209
=CHEDIT  Abs   85145 #14C99 -   130
 CHIRP   Ext                -   245
 CHROUT  Abs   85477 #14DE5 -   236   230
 CKSREQ  Ext                -   584
=CNVUCR  Abs   86695 #152A7 -  1441
=CONVUC  Abs   86698 #152AA -  1442
 CSLW3   Ext                -   491
 CSRW5   Ext                -   963
=CTRL    Abs   86661 #15285 -  1327   509
=CURSFL  Abs   86495 #151DF -  1043   201
=CURSFR  Abs   86487 #151D7 -  1006   202
 CURSOR  Ext                -  1084  1089
 CURSRL  Abs   86479 #151CF -   970   204
=CURSRR  Abs   86471 #151C7 -   967   205
 DO=CUR  Ext                -  1097
 DEBNCE  Ext                -  1504
 DEFADR  Ext                -   134   217   497
 DSPCHA  Ext                -   239
 DSPCHC  Ext                -  1088
 DSPCHR  Abs   85475 #14DE3 -   235   154
 DSPCL?  Ext                -   132
 DSPSPC  Abs   85471 #14DDF -   234   146   150
 ESCSEQ  Ext                -  1221
 ESCSTA  Ext                -   236   240
 FINDAj  Ext                -   503
 FLIPO   Abs   86455 #151B7 -   961   468   952
 FLIPCS  Abs   86424 #15198 -   952   523   948
 FLIPCX  Abs   86457 #151B9 -   962   957   960
 FPOLL   Ext                -   380   419
 FUNCRO  Ext                -   429   436   450   880
 GETDEF  Abs   86403 #15183 -   946   433   482
 I/R10   Abs   86625 #15261 -  1216  1215
 I/R20   Abs   86638 #1526E -  1220  1218
```

```
 IDIVA   Ext                  -  1511
=IN/REP  Abs   86613 #15255 -  1213   198
 Insert  Abs       7 #00007 -    12  1095  1214
 KCOLO   Ext                  -  1513
 KEY100  Abs   85914 #14F9A -   481   478
 KEY110  Abs   85984 #14FE0 -   500   481   483
 KEY120  Abs   86094 #1504E -   536   500
 KEY130  Abs   86121 #15069 -   543   540
 KEYBUF  Ext                  -   541  1505  1537
 KEYCOD  Ext                  -   412   528
 KEYDON  Abs   85966 #14FCE -   497   535   581
 KEYFND  Ext                  -   950
 KEYPTR  Ext                  -   536  1531
 KEYR50  Abs   85528 #14E18 -   380   403   410   457   543
 KEYR60  Abs   85575 #14E47 -   393   399
 KEYR68  Abs   85612 #14E6C -   402   422
 KEYR69  Abs   85619 #14E73 -   405   383
 KEYR70  Abs   85630 #14E7E -   409   394
 KEYR72  Abs   85640 #14E88 -   411   389   408
 KEYR75  Abs   85830 #14F46 -   459   426
 KEYR80  Abs   85875 #14F73 -   471   461   467
 KEYR90  Abs   85906 #14F92 -   479   473
=KEYRD   Abs   85521 #14E11 -   379   133
=KEYTYP  Abs   86293 #15115 -   821   435
=KYTYP1  Abs   86314 #1512A -   829
 LERRM   Ext                  -   761
=LOWERC  Abs   86595 #15243 -  1138   505
 LXTXTT  Ext                  -   568
 MTADDR  Ext                  -   544
 NEEDSC  Ext                  -    57
 NEWT05  Abs   86158 #1508E -   554   552
 NEWT10  Abs   86198 #150B6 -   568   556   566
 NEWTOK  Abs   86125 #1506D -   544   520
=NOSCRL  Abs   85130 #14C8A -    56
 POPBUF  Ext                  -   409
 RANGE   Ext                  -   763
 RCLSTA  Ext                  -   891  1094  1213
 RCURON  Abs   85120 #14C80 -    17   130
 RPTK10  Abs   86737 #152D1 -  1500  1498
 RPTK20  Abs   86749 #152DD -  1503  1550
 RPTK30  Abs   86809 #15319 -  1519  1522
 RPTK40  Abs   86822 #15326 -  1523  1520
 RPTK45  Abs   86846 #1533E -  1530  1524
 RPTK47  Abs   86902 #15376 -  1545  1535
 RPTKCC  Abs   86828 #1532C -  1525  1540  1547
=RPTKY   Abs   86714 #152BA -  1494   386
 RTNCR   Abs   85265 #14D11 -   165   184   186
 SETTMO  Ext                  -   392
 SFLAG?  Ext                  -   764
 SFLAGC  Ext                  -   765
 SFLAGS  Ext                  -  1365
 SFLAGT  Ext                  -  1139
=SFlagC  Abs   86287 #1510F -   765   428   463   475  1526
 SFlagS  Abs   86681 #15299 -  1365  1328  1400  1502
 SLEEP   Ext                  -   393
```

```
=Sflag?  Abs   86281 #15109 -   764   424   460   472   480   522   947  1496
 TBEN    Abs   85389 #14D8D -   206   189
 TBLJMC  Ext               -   195
 TBST    Abs   85359 #14D6F -   196   189
 TIMER1  Ext               -  1530  1587
 TMRRST  Abs   86836 #15334 -  1527  1536
 UDK;    Abs   85313 #14D41 -   183   173   175
=USERK   Abs   86605 #1524D -  1171   507
=USERX   Abs   86669 #1528D -  1362   513
 USRSTA  Ext               -   499
 VIEW    Abs   85696 #14ECO -   427   425
 VIEW20  Abs   85758 #14EFE -   442   440   449
 VIEW30  Abs   85822 #14F3E -   456   444
=VIEWD1  Abs   86343 #15147 -   881
=VIEWK   Abs   86687 #1529F -  1399   511
 VIEWUN  Abs   85780 #14F14 -   450   434
 VWFC-2  Abs   86336 #15140 -   880   456
 WAITKY  Ext               -   895
 WINDLN  Ext               -   881
 WINDST  Ext               -   885
 WIPOUT  Ext               -   432
 WRIT05  Abs   85443 #14DC3 -   224   163
 WRIT10  Abs   85469 #14DDD -   232   229
 WRITE   Abs   85406 #14D9E -   213   147   183   210   226   231
=WRTTM1  Abs   86923 #1538B -  1587  1500  1529
=WRTTMR  Abs   86930 #15392 -  1588
 WRTTMX  Abs   86949 #153A5 -  1594  1592
=aRANGE  Abs   86269 #150FD -   762   959  1442
=bf2dsp  Abs   86582 #15236 -  1103   131
 cksreq  Abs   86241 #150E1 -   583   396  1549
 escseq  Abs   86638 #1526E -  1221   968   971  1007  1044  1256  1292
 fTMOUT  Abs       3 #00003 -    14   398
 flCTRL  Ext               -   459   462  1327
 flLC    Ext               -   521   946  1138
 flRPTD  Ext               -  1495  1501  1525
 flUSER  Ext               -   476   479  1171  1362
 flUSRX  Ext               -   471   474  1364
 flVIEW  Ext               -   423   427  1399
 kWOFF   Ext               -   400
 kcCTRL  Ext               -   508
 kcLC    Ext               -   504
 kcLERR  Ext               -   514
 kcUSER  Ext               -   506
 kcUSEX  Ext               -   512
 kcVIEW  Ext               -   510
=keyfnd  Abs   86417 #15191 -   950
 lerrm   Abs   86262 #150F6 -   761   515
 pKYDF   Ext               -   420
 pWTKY   Ext               -   381
 range   Abs   86275 #15103 -   763   466
 rstlvl  Ext               -   585
 savlvl  Ext               -   583
 sflagt  Abs   86599 #15247 -  1139   477  1172  1363
 usrsta  Abs   85977 #14FD9 -   499   390   395  1548
=waitky  Abs   86397 #1517D -   895
```

Input Parameters

  Source file name is MN&ED::MS

  Listing file name is MN/ED:TI:ML::-1

  Object file name is MN%ED:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
1                    TITLE  Zero File - End of chain
2 00000 00           NIBHEX 00
3 00002              END
```

Input Parameters

  Source file name is JP&ZER::MS

  Listing file name is JP/ZER:TI:ML::-1

  Object file name is JP%ZER:TI:MS::-1

```
                                  111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    TTTTT  III   &     RRRR   EEEEE  V   V
 2            *      T     I   & &    R   R  E      V   V
 3            *      T     I   & &    R   R  E      V   V
 4            *      T     I    &     RRRR   EEEE    V V
 5            *      T     I   & & &  R R    E       V V
 6            *      T     I   &  &   R  R   E        V
 7            *      T    III   && &  R   R  EEEEE    V
 8
 9                      TITLE  HP-71 Revision Number
10 00000 24             NIBASC \B\
11 00002                END
```

Input Parameters

Source file name is TI&REV::MS

Listing file name is TI/REV:TI:ML::-1

Object file name is TI%REV:TI:MS::-1

```
                                111111
                       0123456789012345
```

Initial flag settings are

Errors

None

Saturn Assembler News

```
 1            *    TTTTT  III   &     FFFFF  X   X   333
 2            *      T     I   & &    F      X   X  3   3
 3            *      T     I   & &    F       X X       3
 4            *      T     I    &     FFFF     X     333
 5            *      T     I   & & &  F       X X       3
 6            *      T     I   &  &   F      X   X  3   3
 7            *      T    III   && &  F      X   X   333
 8            *
 9                        TITLE ROM 3 Fix Module
10 153AD                  ABS    #153AD
11 153AD                  END
```

Input Parameters

  Source file name is TI&FX3::MS

  Listing file name is TI/FX3:TI:ML::-1

  Object file name is TI%FX3:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *      A    BBBB    &      CCC   L       CCC
 2            *     A A   B   B  & &    C   C  L      C   C
 3            *    A   A  B   B  & &    C      L      C
 4            *    A   A  BBBB    &     C      L      C
 5            *    AAAAA  B   B  & & &  C      L      C
 6            *    A   A  B   B  &  &   C   C  L      C   C
 7            *    A   A  BBBB    && &   CCC   LLLLL   CCC
 8
 9                   TITLE  CALC Mode <831228.1215>
10 1551F            ABS    #1551F
11
12                   RDSYMB SB%TAB::MS
13
14
15           ********************************************************************
16           **
17           **                  CALC MODE STATE TABLE
18           **
19           **   The following state machine approximates the operation of
20           **   the HP-71 CALC Mode.
21           **
22           **
23           **    STATE 0:OPERAND EXPECTED
24           **
25           **          +            NOP.
26           **          BLANK
27           **
28           **          -            TOKEN EQL TOP(RSTACK)
29           **          NOT           THEN
30           **                         RSTACK
31           **                        ELSE
32           **                         TOKEN->RSTACK.
33           **
34           **
35           **          FUNCTION     LPAREN
36           **                        THEN
37           **                         MAXARG
38           **                          THEN
39           **                           BEGIN
40           **                            TOKEN->RSTACK;
41           **                            "("->RSTACK;
42           **                            RUN INSERT(")");
43           **                            LEXPTR+2->LEXPTR;
44           **                            INDEX COMMA FROM MINARG TO 2 BY -1:
45           **                             RUN INSERT(",")
46           **                           END
47           **                          ELSE
48           **                           RUN ERROR
49           **                        ELSE
50           **                         MINARG
51           **                          THEN
52           **                           RUN ERROR
53           **                          ELSE
54           **                           BEGIN
55           **                            TOKEN->RSTACK;
```

```
                **                          1->STATE
                **                         END.
                **
                **
                **        NUMBER           NUMBER->NSTACK;
                **        OR               1->STATE.
                **        VARIABLE
                **
                **        (                LPAREN->RSTACK;
                **                         RUN INSERT(RPAREN).
                **
                **
                **        ,                CURCHAR(COMMA)
                **                          THEN
                **                           BEGIN
                **                            COMMA->RSTACK;
                **                            RESULTREGISTER->NSTACK
                **                           END
                **                          ELSE
                **                           CURCHAR(RPAREN)
                **                            THEN
                **                             BEGIN
                **                              RUN INSERT(COMMA);
                **                              COMMA->RSTACK;
                **                              RESULTREGISTER->NSTACK
                **                             END
                **                            ELSE
                **                             RUN ERROR.
                **
                **
                **        )                CURCHAR(COMMA)
                **                          THEN
                **                           BEGIN
                **                            COMMA->RSTACK;
                **                            RESULTREGISTER->NSTACK
                **                           END
                **                          ELSE
                **                           CURCHAR(RPAREN)
                **                            THEN
                **                             BEGIN
                **                              RPAREN->RSTACK;
                **                              RESULTREGISTER->NSTACK;
                **                              1->STATE
                **                             END
                **                            ELSE
                **                             RUN ERROR.
                **
                **
                **        ALL OTHER INPUTS ARE ERRORS. RUN ERROR.
                **
                **
                **    STATE 1: OPERATOR EXPECTED
                **
                **        EOL              RUN EXPR;
                **                         NSTACK->RESULTREGISTER;
```

```
111          **                    ASSIGNMENT
112          **                     THEN
113          **                      BEGIN
114          **                       RSTACK;
115          **                       RUN EXPR;
116          **                       RUN DEST;
117          **                       RESULTREGISTER->NSTACK;
118          **                       RUN STORE
119          **                      END;
120          **                    STOP.
121          **
122          **          -         REPEAT
123          **                     PRECEDENCE(PLUS,TOP(RSTACK))
124          **                      THEN
125          **                       PLUS->RSTACK EXIT
126          **                      ELSE
127          **                       RUN OPERATE;
128          **                    MINUS->RSTACK;
129          **                    0->STATE.
130          **
131          **
132          **          BINARY    REPEAT
133          **          OPERATOR   PRECEDENCE(TOKEN,TOP(RSTACK))
134          **                      THEN
135          **                       TOKEN->RSTACK EXIT
136          **                      ELSE
137          **                       RUN OPERATE;
138          **                    0->STATE.
139          **
140          **
141          **          ,         CURCHAR(COMMA)
142          **                     THEN
143          **                      BEGIN
144          **                       RUN EXPR;
145          **                       COMMA->RSTACK
146          **                      END
147          **                     ELSE
148          **                      CURCHAR(RPAREN)
149          **                       THEN
150          **                        BEGIN
151          **                         RUN EXPR;
152          **                         RUN INSERT(COMMA);
153          **                         COMMA->RSTACK
154          **                        END
155          **                       ELSE
156          **                        RUN ERROR;
157          **                    0->STATE.
158          **
159          **
160          **          )         CURCHAR(COMMA)
161          **                     THEN
162          **                      BEGIN
163          **                       RUN EXPR;
164          **                       COMMA->RSTACK;
165          **                       0->STATE
```

```
166             **                        END
167             **                       ELSE
168             **                        CURCHAR(RPAREN)
169             **                         THEN
170             **                          BEGIN
171             **                           RUN EXPR;
172             **                           RPAREN->RSTACK
173             **                          END
174             **                         ELSE
175             **                          RUN ERROR.
176             **
177             **
178             **          ALL OTHER INPUTS ARE ERRORS. RUN ERROR.
179             **
180             **
181             **     STATE 2: BLANK LAST ENTRY
182             **
183             **          BLANK          TOP(RSTACK) EQL COMMA
184             **                         OR TOP(RSTACK) EQL LPAREN
185             **                          THEN
186             **                           RUN ERROR
187             **                          ELSE
188             **                           BEGIN
189             **                            PRECEDENCE(LIMIT,TOP(RSTACK))
190             **                             THEN
191             **                              TOP(RSTACK)->LIMIT;
192             **                            RUN OPERATE
193             **                           END.
194             **
195             **
196             **          -              PRECEDENCE(PLUS,LIMIT)
197             **                          THEN
198             **                           BEGIN
199             **                            RUN ERROR;
200             **                            RUN ADVANCE
201             **                           END
202             **                          ELSE
203             **                           1->STATE.
204             **
205             **          BINARY         PRECEDENCE(TOKEN,LIMIT)
206             **          OPERATOR        THEN
207             **                           BEGIN
208             **                            RUN ERROR;
209             **                            RUN ADVANCE
210             **                           END
211             **                          ELSE
212             **                           1->STATE.
213             **
214             **          ,              1->STATE.
215             **          )
216             **          EOL
217             **
218             **          ALL OTHER INPUTS ARE ERRORS. RUN ERROR.
219             **
220             **
```

```
221             **    CALC MODE DECOMPILE
222             **
223             **                        RUN DUMPBACK;
224             **                        STATE EQL 2
225             **                         THEN
226             **                          RUN DISPLAYNUMBER;
227             **                        REPEAT
228             **                         BEGIN
229             **                          REPEAT
230             **                           TOP(RSTACK) EQL EOL
231             **                            THEN
232             **                             EXIT REPEAT REPEAT
233             **                            ELSE
234             **                             TOP(RSTACK) IN FUNCTIONSET
235             **                              THEN
236             **                               RUN DISPLAYOPERATOR
237             **                              ELSE
238             **                               BEGIN
239             **                                RUN DISPLAYOPERATOR;
240             **                                EXIT REPEAT
241             **                               END;
242             **                         RUN DISPLAYNUMBER
243             **                        END.
244             **
245             ****************************************************************
```

```
246                     EJECT
247             ********************************************************************
248             **
249             **      STATUS BIT ASSIGNMENTS
250             **
251             ********************************************************************
252
253              BEGIN  EQU    0              1=>CHECK FOR BEGIN BASIC TOKEN
254              STRING EQU    1              1=>STRING
255              ARG    EQU    2              1=>ARGUMENT LIST PRESENT
256              NOP    EQU    3              1=>SIGN CHARACTER IS NOT OPERATOR
257              DSTATE EQU    4              DECOMPILE STATE
258              RSTATE EQU    5              REDUCTION STATE
259              LOCAL  EQU    6              1=>NON-NULL LOCAL EXPRESSION
260              ASSIGN EQU    7              1=>ASSIGNMENT STATEMENT
261              BASIC  EQU    8              1=>NON-ASSIGNMENT STATEMENT
262              SNAG   EQU    9              1=>EOL IS NOP
263              PAREN  EQU    10             1=>STRICT PARENTHESIS MATCHING
264              VAR    EQU    11             1=>VARIABLE
```

```
265                     EJECT
266                 ****************************************************************
267                 ****************************************************************
268                 **
269                 ** Name:    STKEND  -  Locate End Of Command STACK
270                 **
271                 ** Category:   LOCAL
272                 **
273                 ** Purpose:
274                 **      To locate the end of the Command Stack.
275                 **
276                 ** Entry:
277                 **      HEXMODE
278                 **
279                 ** Exit:
280                 **      A(A)   =  Address of end of Command Stack
281                 **      Carry Clear
282                 **
283                 ** Calls:      Nothing
284                 **
285                 ** Uses.....  D0, C(S), C(A), A(A)
286                 **
287                 ** Stk lvls:   0
288                 **
289                 ** History:
290                 **
291                 **    Date     Programmer              Modification
292                 **  --------   ----------   ---------------------------------
293                 **  06/09/83   SA           Added documentation
294                 **
295                 ****************************************************************
296                 ****************************************************************
297 1551F 1B00 =STKEND D0=(5) =MAXCMD
            000
298 15526 1564          C=DAT0 S
299 1552A 1A00          D0=(4) =CLCBFR
            00
300 15530 142           A=DAT0 A
301 15533 D2            C=0    A
302 15535 132   SE10    ADOEX
303 15538 1563          C=DAT0 X
304 1553C 165           D0=D0+ 6
305 1553F 132           ADOEX
306 15542 CA            A=A+C  A
307 15544 A4E           C=C-1  S
308 15547 5DE           GONC   SE10
309 1554A 03            RTNCC
310                 ****************************************************************
311                 ****************************************************************
312                 **
313                 ** Name:    SYSTEM  -  InitializE CALC MODE
314                 **
315                 ** Category:   LOCAL
316                 **
317                 ** Purpose:
```

```
318            **      To initialize CALC Mode.
319            **
320            ** Entry:
321            **      None
322            **
323            ** Exit:
324            **      See NEWSTM
325            **
326            ** Calls:       CRLFOF, CLRXDS, DSPRST, SFLAGS, CLCCFR, STKEND,
327            **              NEWSTM
328            **
329            ** Uses....... Everything
330            **
331            ** Stk lvls:   xxxxxxx
332            **
333            ** History:
334            **
335            **    Date      Programmer              Modification
336            **  --------    ----------   -----------------------------------
337            **  06/09/83    SA           Added documentation
338            **
339            ************************************************************************
340            ************************************************************************
341 1554C 04   =SYSTEM SETHEX
342 1554E 8E96         GOSUBL KILLKY          KILL KEY VECTOR
           80
343 15554 8F00         GOSBVL =CRLFOF         TURN OFF CURSOR
           000
344 1555B 8F00         GOSBVL =CLRXDS         DISABLE EXTERNAL DISPLAY
           000
345 15562 8F00         GOSBVL =DSPRST         RESET DISPLAY
           000
346 15569 3100         LC(2)  =flCALC
347 1556D 8E00         GOSUBL =SFLAGS         SET CALC FLAG
           00
348 15573 7EB3         GOSUB  clccfr          INITIALIZE DISPLAY PARAMETERS
349 15577 74AF         GOSUB  STKEND          FIND END OF COMMAND STACK
350 1557B 131          D1=A
351 1557E 6700         GOTO   NEWSTM          GOTO NEWSTM
352            ************************************************************************
353            ************************************************************************
354            **
355            ** Name:    PSHSTM  -  Push Statement On Cmd StacK, INIT CALC
356            ** Name:    NEWSTM  -  InitiALIZE CALC
357            **
358            ** Category:   LOCAL
359            **
360            ** Purpose:
361            **      PSHSTM:
362            **        Pushes statement on command stack, initializes CALC
363            **      NEWSTM:
364            **        Initializes CALC
365            ** Entry:
366            **      P      =  0
367            **
```

```
368            ** Exit:
369            **      P       =  0
370            **
371            ** Calls:       STKCMD
372            **
373            ** Uses.......  A, B, C, D, P, D0, D1
374            **
375            ** Stk lvls:    2 for PSHSTM, 0 for NEWSTM
376            **
377            ** History:
378            **
379            **    Date      Programmer              Modification
380            **  --------    ----------   ----------------------------------
381            **  06/09/83    SA           Added documentation
382            **
383            ****************************************************************
384            ****************************************************************
385 15582 7760 =PSHSTM GOSUB  STKCMD          PUSH STATEMENT ON COMMAND STACK
386 15586 3410 =NEWSTM LCHEX  00001
          000
387 1558D 1B00          D0=(5) =S-R1-2        INITIALIZE STATEMENT LENGTH,
          000
388 15594 144           DAT0=C A               CONTEXT LIMIT, & SST COUNT
389 15597 3300          LCHEX  D000
          0D
390 1559D 145           DAT1=C A              INITIALIZE NEW STATEMENT
391 155A0 172           D1=D1+ 3
392 155A3 133           AD1EX
393 155A6 1F00          D1=(5) =RFNBFR
          000
394 155AD 141           DAT1=A A              INITIALIZE RFNBFR
395 155B0 174           D1=D1+ 5
396 155B3 141           DAT1=A A              INITIALIZE RAWBFR
397 155B6 E4            A=A+1  A
398 155B8 E4            A=A+1  A
399 155BA 174           D1=D1+ 5
400 155BD 141           DAT1=A A              INITIALIZE CLCSTK
401 155C0 1D00          D1=(2) =FORSTK
402 155C4 143           A=DAT1 A              READ FORSTK
403 155C7 1C4           D1=D1- 5
404 155CA 141           DAT1=A A              INITIALIZE MTHSTK
405 155CD 133           AD1EX
406 155D0 1C3           D1=D1- 4
407 155D3 3311          LC(4)  (tEOL)*#100+#11
          0F
408 155D9 15D3          DAT1=C 4
409 155DD 133           AD1EX
410 155E0 1C4           D1=D1- 5
411 155E3 141           DAT1=A A              INITIALIZE AVMEMS
412 155E6 08            CLRST                 INITIALIZE STATUS
413            *        ST=1   BEGIN
414 155E8 854           ST=1   DSTATE
415 155EB 03            RTNCC                 RETURN CARRY CLEAR
416            ****************************************************************
417            ****************************************************************
```

```
418                **
419                ** Name:(S) STKCMD  -  Pushes Statement On Command STACK
420                **
421                ** Category:   GENUTL
422                **
423                ** Purpose:
424                **      Pushes statement on command stack.
425                **
426                ** Entry:
427                **      P      =  0
428                **
429                ** Exit:
430                **      P      =  0
431                **
432                ** Calls:      ORGN10, STREQL, MOVEU3
433                **
434                ** Uses.....  A, B, C, D, P, D0, D1
435                **
436                ** Stk lvls:   1
437                **
438                ** History:
439                **
440                **    Date     Programmer             Modification
441                **  --------   ----------   ---------------------------------
442                **  06/09/83   SA           Added documentation
443                **
444                ********************************************************************
445                ********************************************************************
446 155ED 1F00 =STKCMD D1=(5) =MAXCMD
           000
447 155F4 1534          A=DAT1 S              READ COMMAND STACK DEPTH
448 155F8 AC8           B=A    S
449 155FB 1E00          D1=(4) =RAWBFR
           00
450 15601 7E62          GOSUB  ORGN10         COUNT CHARACTERS IN LINE
451 15605 CF            D=D-1  A
452 15607 C7            D=D+D  A              CONVERT BYTES TO NIBBLES
453 15609 D6            C=A    A
454 1560B 134           D0=C
455 1560E DF            CDEX   A
456 15610 1553          DAT1=C X              WRITE LENGTH TO START OF LINE
457 15614 E6            C=C+1  A
458 15616 E6            C=C+1  A
459 15618 E6            C=C+1  A
460 1561A 1543          DAT0=C X              WRITE LENGTH TO END OF LINE
461 1561E D5            B=C    A
462 15620 816           CSRC
463 15623 1C4           D1=D1- 5
464 15626 14B           A=DAT1 B              READ LAST CHARACTER OF LAST LINE
465 15629 34D0          LCHEX  0000D
           000
466 15630 962           ?A=C   B              ENDLINE?
467 15633 50            GOYES  PS05           IF SO, GOTO PS05
468 15635 AC1           B=0    S              SET STACK COUNTER TO ZERO
469 15638 174   PS05    D1=D1+ 5
```

```
470 1563B 80DF          P=C    15          SET POINTER FOR STRING COMPARISON
471 1563F 0D            P=P-1
472 15641 1C2   PS10    D1=D1- 3
473 15644 1573          C=DAT1 X           READ PREVIOUS STATEMENT LENGTH
474 15648 935           ?B#C   X           EQUAL TO NEW STATEMENT LENGTH?
475 1564B 13            GOYES  PS30        IF NOT, GOTO PS30
476 1564D A3D           B=B-1  X
477 15650 BB5           BSR    X
478 15653 8E00          GOSUBL =STREQL     COMPARE STRINGS
          00
479 15659 D2            C=0    A
480 1565B 5E2           GONC   PS40        IF EQUAL, GOTO PS40
481 1565E E5            B=B+1  A           COMPUTE REMAINING LENGTH TO START
482 15660 4D0           GOC    PS20
483 15663 CD            B=B-1  A
484 15665 BB1           BSL    X
485 15668 AB9           C=B    X
486 1566B 809           C+P+1
487 1566E DF    PS20    CDEX   A
488 15670 134           D0=C               RESET D0 TO START OF NEW STMT
489 15673 DF            CDEX   A
490 15675 1523          A=DAT0 X
491 15679 AB8           B=A    X           RESET NEW STATEMENT LENGTH
492 1567C 133   PS30    AD1EX
493 1567F EA            A=A-C  A           COMPUTE ADDR OF HIGHER STACK ELT
494 15681 133           AD1EX
495 15684 A4D           B=B-1  S           STACK TRAVERSED?
496 15687 59B           GONC   PS10        IF NOT, REPEAT PS10
497 1568A 1573  PS40    C=DAT1 X
498 1568E 133           AD1EX
499 15691 C2            C=A+C  A           COMPUTE ADDR OF LOWER STACK ELT
500 15693 22            P=     2
501 15695 809           C+P+1
502 15698 134           D0=C
503 1569B 162           D0=D0+ 3
504 1569E 131           D1=A
505 156A1 DF            CDEX   A
506 156A3 EB            C=C-D  A           COMPUTE LENGTH OF STACK ELEMENT
507 156A5 8C00  =Moveu3 GOLONG =MOVEU3     DELETE STACK ELEMENT
          00
508             ****************************************************************
509             ****************************************************************
510             **
511             ** Name:    KEYCHR  -  Gets Next Char Of Key DefinITION
512             **
513             ** Category:   LOCAL
514             **
515             ** Purpose:
516             **      Gets next character in key definition, supplies EOL or
517             **      left parenthesis if necessary.
518             **
519             ** Entry:
520             **      P      =  0
521             **
522             ** Exit:
```

```
523            **      P      =  0
524            **      Modifies DEFADR: decrements length field, increments
525            **        address.
526            **      Carry Set => A(B) = Next character in key definition.
527            **      Carry Clear=>Key definition exhausted.
528            **
529            ** Calls:
530            **
531            ** Uses..... A, C, D0
532            **
533            ** Stk lvls:   0
534            **
535            ** History:
536            **
537            **    Date      Programmer              Modification
538            **  --------    ----------   ------------------------------------
539            **  06/09/83    SA           Added documentation
540            **
541            ********************************************************************
542            ********************************************************************
543 156AB 1B00  KEYCHR D0=(5) =DEFADR
          000
544 156B2 1523          A=DAT0 X              READ DEFINITION INFORMATION
545 156B6 A6C           A=A-1  B              STRING EXHAUSTED?
546 156B9 4E1           GOC    ENDDEF         IF SO, GOTO ENDDEF
547 156BC 148           DAT0=A B              STORE BYTES REMAINING
548 156BF 162           D0=D0+ 3
549 156C2 146           C=DAT0 A              READ DEFINITION ADDRESS
550 156C5 136           CDOEX
551 156C8 D0            A=0    A
552 156CA 14A           A=DAT0 B              READ NEXT CHARACTER
553 156CD 161           D0=D0+ 2
554 156D0 136           CDOEX
555 156D3 144           DAT0=C A              STORE ADDRESS
556 156D6 02            RTNSC                 RETURN CARRY SET
557
558 156D8 84A   ENDDEF  ST=0   PAREN
559 156DB AB2           C=0    X
560 156DE 1543          DAT0=C X              SET TERMINAL STATUS
561 156E2 A24           A=A+A  XS             LEX TABLE ENTRY?
562 156E5 5D1           GONC   END30          IF NOT, GOTO END30
563 156E8 A24           A=A+A  XS
564 156EB 3102          LCASC  \ \            LOAD BLANK
565 156EF A24           A=A+A  XS             REQUIRED?
566 156F2 490           GOC    END20          IF SO, GOTO END20
567 156F5 3182          LCASC  \(\            LOAD LEFT PARENTHESIS
568 156F9 A24   END10   A=A+A  XS             REQUIRED?
569 156FC D0    END20   A=0    A
570 156FE AEA           A=C    B
571 15701 01            RTN                   IF SO, SET CARRY & RETURN
572
573 15703 A24   END30   A=A+A  XS             NONTERMINATING USERKEY?
574 15706 4A0           GOC    END40          IF SO, GOTO END40
575 15709 31D0          LCHEX  0D             LOAD CARRIAGE RETURN
576 1570D 6BEF          GOTO   END10          GOTO END10
```

```
577
578 15711 03    END40  RTNCC                  RETURN CARRY CLEAR
579             ******************************************************************
580             ******************************************************************
581             **
582             ** Name:    BUFFER  -  Insert CHAR AT RAWBFR
583             **
584             ** Category:   LOCAL
585             **
586             ** Purpose:
587             **      Insert char at RAWBFR, or jump to CALC Edit Routines.
588             **
589             ** Entry:
590             **      P      =  0
591             **      A(B) = Character to be inserted or jumped on.
592             **      Carry Clear => Jump into Command Stack because of
593             **      unrecoverable error.
594             **
595             ** Exit:
596             **      P      =  0
597             **      Carry Set => B(B) = Character inserted at RAWBFR, which
598             **      may have been moved.
599             **      Carry Clear => Something Strange happened. Please note
600             **      that, of those CALC Edit Routines which return, most do
601             **      so with Carry Clear. See CALC for more information.
602             **
603             ** Calls:      CLCCFR, ERRMSG, HASH2 (and the whole CALC Edit
604             **             system), MEMBER, INSRT1
605             **
606             ** Uses....... Everything
607             **
608             ** Stk lvls:   6
609             **
610             ** History:
611             **
612             **    Date     Programmer              Modification
613             **  --------   ----------   ----------------------------------
614             **  06/09/83   SA           Added documentation
615             **
616             ******************************************************************
617             ******************************************************************
618 15713 506   BUFFER GONC   FORCE          IF CARRY CLEAR, GOTO FORCE
619 15716 31A1         LCHEX  1A
620 1571A 9E2          ?A<C   B              CONTROL CHARACTER?
621 1571D 06           GOYES  CTRL           IF SO, GOTO CTRL
622 1571F 7212         GOSUB  clccfr         RESET DISPLAY PARAMETERS
623 15723 AE8          B=A    B              MOVE INPUT CHARACTER TO B
624 15726 1B00         D0=(5) (=S-R1-2)+4
          000
625 1572D A82          C=0    P
626 15730 1540         DAT0=C P              KILL SST COUNT
627 15734 1A00         D0=(4) =RAWBFR
          00
628 1573A 142          A=DAT0 A
629 1573D 131          D1=A
```

```
630 15740 14B          A=DAT1 B            READ CHARACTER AT RAWBFR
631 15743 3192         LCASC  \)\
632 15747 965          ?B#C   B            RIGHT PARENTHESIS GIVEN?
633 1574A 04           GOYES  BFR40        IF NOT, GOTO BFR40
634 1574C 962          ?A=C   B            RIGHT PARENTHESIS EXPECTED?
635 1574F 01           GOYES  BFR30        IF SO, GOTO BFR30
636 15751 87A          ?ST=1  PAREN        STRICT PARENTHESIS MATCHING ON?
637 15754 61           GOYES  BFR55        IF SO, GOTO BFR55
638 15756 31C2         LCASC  \,\
639 1575A 966          ?A#C   B            COMMA EXPECTED?
640 1575D DO           GOYES  BFR55        IF NOT, GOTO BFR55
641 1575F 171  BFR30   D1=D1+ 2            ADVANCE RAWBFR
642 15762 137          CD1EX
643 15765 144          DAT0=C A
644 15768 02           RTNSC               RETURN CARRY SET
645
646 1576A 03   BFR55   RTNCC               RETURN CARRY CLEAR
647
648 1576C 3100 BFR60   LC(2)  =eL2LNG
649 15770 77C1         GOSUB  ERRMSG
650 15774 3100 FORCE   LC(2)  =kcBOT        FORCE EDIT
651 15778 DO           A=0    A
652 1577A AEA          A=C    B
653 1577D 3400 CTRL    LC(5)  =CLCET1
          000
654 15784 8C00         GOLONG =HASH2        JUMP TO EDIT FUNCTION
          00
655
656 1578A 31C2 BFR40   LCASC  \,\
657 1578E 965          ?B#C   B            COMMA GIVEN?
658 15791 01           GOYES  BFR50        IF NOT, GOTO BFR50
659 15793 962          ?A=C   B            COMMA EXPECTED?
660 15796 9C           GOYES  BFR30        IF SO, GOTO BFR30
661 15798 3192         LCASC  \)\
662 1579C 966          ?A#C   B            RIGHT PARENTHESIS EXPECTED?
663 1579F BC           GOYES  BFR55        IF NOT, GOTO BFR55
664 157A1 AE4  BFR50   A=B    B
665 157A4 3D06         LCASC  \:;@[]_`\
          F5D5
          B504
          B3A3
666 157B4 2D           P=     13
667 157B6 8E00         GOSUBL =MEMBER
          00
668 157BC 5B0          GONC   BFR51        TRANSLATE ILLEGAL CHARACTERS TO !
669 157BF 31B7         LCASC  \{\
670 157C3 9E2          ?A<C   B
671 157C6 90           GOYES  BFR52
672 157C8 3112 BFR51   LCASC  \!\
673 157CC AE5          B=C    B
674 157CF 7130 BFR52   GOSUB  INSRT1        INSERT CHARACTER AT RAWBFR
675 157D3 489          GOC    BFR60        IF INSERTION FAILS, GOTO BFR60
676 157D6 145          DAT1=C A            UPDATE RAWBFR
677 157D9 02           RTNSC               RETURN CARRY SET
678          ****************************************************************
```

```
679            ****************************************************************
680            **
681            ** Name:    ACTIVE  -  Check Need For CALC ReduCTION
682            **
683            ** Category:   LOCAL
684            **
685            ** Purpose:
686            **      Determine whether to activate CALC Mode Reduction
687            **      Engine.
688            **
689            ** Entry:
690            **      P      =  0
691            **      B(B)   =  Character to perform activation test on.
692            **      Active characters are ^, /, and anything less than
693            **      ASCII period.
694            **
695            ** Exit:
696            **      P      =  0
697            **      Carry Set => Active character.
698            **
699            ** Calls:
700            **
701            ** Uses....... B(B), C(B)
702            **
703            ** Stk lvls:   0
704            **
705            ** History:
706            **
707            **    Date     Programmer             Modification
708            **  --------   ----------   ---------------------------------
709            **  06/09/83   SA           Added documentation
710            **  07/08/83   SA           Deleted # from activation set
711            **
712            ****************************************************************
713            ****************************************************************
714 157DB 31E5  ACTIVE LCASC  \^\
715 157DF 961          ?B=C   B              INVOLUTION OPERATOR?
716 157E2 00           RTNYES                IF SO, RETURN CARRY SET
717 157E4 31F2         LCASC  \/\
718 157E8 961          ?B=C   B              DIVISION OPERATOR?
719 157EB 00           RTNYES                IF SO, RETURN CARRY SET
720 157ED 3132         LCASC  \#\
721 157F1 961          ?B=C   B              NOTEQUAL OPERATOR?
722 157F4 B0           GOYES  ACTV10         IF SO, RETURN CARRY CLEAR
723 157F6 31E2         LCASC  \.\
724 157FA 9E5          ?B<C   B              OTHER ACTIVATION CHARACTER?
725 157FD 00           RTNYES                IF SO, RETURN CARRY SET
726 157FF 03    ACTV10 RTNCC                 RETURN CARRY CLEAR
727            ****************************************************************
728            ****************************************************************
729            **
730            ** Name:    INSRT0  -  Insert A CharacTER AT RAWBFR
731            ** Name:    INSRT1  -  Insert A CharacTER AT RAWBFR
732            **
733            ** Category:   LOCAL
```

```
734                **
735                ** Purpose:
736                **      Insert character at RAWBFR
737                **
738                ** Entry:
739                **      P      = 0
740                **      For INSRT0, character in C(B)
741                **      For INSRT1, character in B(B)
742                **
743                ** Exit:
744                **      P      = 0
745                **      Carry Set => Insufficient Memory or Line Too Long.
746                **      Carry Clear =>
747                **      D1     = RAWBFR
748                **      C(A)   = New value for RAWBFR, if it needs moving.
749                **      B(B)   = Character inserted.
750                **
751                ** Calls:      MOVED1
752                **
753                ** Uses....... A, B, C, D, P, D0, D1
754                **
755                ** Stk lvls:   1
756                **
757                ** History:
758                **
759                **    Date     Programmer             Modification
760                **  --------   ----------   ----------------------------------
761                **  06/10/83   SA           Added documentation
762                **
763                ****************************************************************
764                ****************************************************************
765 15801 AE5    INSRT0 B=C    B               SAVE CHARACTER IN B
766 15804 3406   INSRT1 LCHEX  00060           LOAD COMPARISON VALUE
          000
767 1580B AE7           D=C    B
768 1580E 1F00          D1=(5) =S-R1-2
          000
769 15815 14F           C=DAT1 B               READ CURRENT LENGTH
770 15818 B66           C=C+1  B               INCREMENT LENGTH
771 1581B 9E7           ?C>D   B               TOO LONG?
772 1581E 00            RTNYES                 IF SO, RETURN CARRY SET
773 15820 AEF           CDEX   B
774 15823 A66           C=C+C  B
775 15826 1B00          D0=(5) =AVMEMS
          000
776 1582D 142           A=DAT0 A               READ AVMEMS
777 15830 EE            C=A-C  A
778 15832 18E           D0=D0- 15
779 15835 142           A=DAT0 A               READ CLCSTK
780 15838 E4            A=A+1  A
781 1583A E4            A=A+1  A
782 1583C E2            C=C-A  A               TOO LITTLE FREE SPACE?
783 1583E 400           RTNC                   IF SO, RETURN CARRY SET
784 15841 AEB           C=D    B
785 15844 14D           DAT1=C B               UPDATE STATEMENT LENGTH COUNT
```

```
786 15847 140          DATO=A A              UPDATE CLCSTK
787 1584A 184          DO=DO- 5
788 1584D 131          D1=A
789 15850 CC           A=A-1  A
790 15852 CC           A=A-1  A
791 15854 8E00         GOSUBL =MOVED1        OPEN RAW BUFFER
          00
792 1585A AE4          A=B    B
793 1585D 148          DATO=A B              INSERT NEW CHARACTER
794 15860 137          CD1EX                 PREPARE FOR RAWBFR UPDATE
795 15863 1F00         D1=(5) =RAWBFR
          000
796 1586A 03           RTNCC                 RETURN CARRY CLEAR
797              ****************************************************************
798              ****************************************************************
799              **
800              ** Name:    ORIGIN  -  Find Start Of Cmd Stk ELEMENT, CNTCHR
801              ** Name:    CNTCHR  -  Count Characters In ELEMENT
802              **
803              ** Category:   LOCAL
804              **
805              ** Purpose:
806              **      Locate start of current Command Stack element; count
807              **      characters in the element.
808              **
809              ** Entry:
810              **      P      =  0
811              **      For ORIGIN, backscan starts at RFNBFR.
812              **      For ORGN10, backscan starts at pointer pointed at by
813              **        D1.
814              **      For CNTCHR, backscan starts at pointer in A(A).
815              **
816              ** Exit:
817              **      P      =  0
818              **      D1     @  Start of current Command Stack element.
819              **      D(B)   =  Number of characters + 1
820              **
821              **
822              **
823              ** Calls:
824              **
825              ** Uses....... A, C, D, D1
826              **
827              ** Stk lvls:   0
828              **
829              ** History:
830              **
831              **    Date     Programmer             Modification
832              **  --------   ----------   ------------------------------------
833              **  06/10/83   SA           Added documentation
834              **
835              ****************************************************************
836              ****************************************************************
837 1586C 1F00 =ORIGIN D1=(5) =RFNBFR
          000
```

```
 838 15873 143   ORGN10 A=DAT1 A
 839 15876 131  =CNTCHR D1=A
 840 15879 D3           D=0    A
 841 1587B B67   BACKUP D=D+1  B
 842 1587E 1C1          D1=D1- 2
 843 15881 14F          C=DAT1 B
 844 15884 96E          ?C#0   B
 845 15887 4F           GOYES  BACKUP
 846 15889 1C0          D1=D1- 1
 847 1588C 03           RTNCC
 848             ********************************************************************
 849             ********************************************************************
 850             **
 851             ** Name:    CALC    -  Entry Point FoR CALC MODE
 852             **
 853             ** Category:   SYSTEM
 854             **
 855             ** Purpose:
 856             **      This is CALC Mode.
 857             **
 858             ** Entry:
 859             **      NONE
 860             **
 861             ** Exit:
 862             **      NONE
 863             **
 864             ** Calls:      Everything in ABCLC, ABBLD, ABED
 865             **
 866             ** Uses..... Everything
 867             **
 868             ** Stk lvls:   7
 869             **
 870             ** History:
 871             **
 872             **    Date     Programmer             Modification
 873             **  --------   ----------   ----------------------------------
 874             **  06/10/83   SA           Added documentation
 875             **
 876             ********************************************************************
 877             ********************************************************************
 878 1588E 7ABC =CALC   GOSUB  SYSTEM         CREATE CALC MODE SYSTEM
 879 15892 8F00         GOSBVL =TNOFF?       CHECK TURNOFF FLAG
           000
 880 15899 3400         LC(5)  =kcOFF
           000
 881 158A0 DA           A=C    A
 882 158A2 461          GOC    EDITOR
 883 158A5 8E00  DSPLAY GOSUBL =BUILD        DECOMPILE REDUCED EXPRESSION
           00
 884 158AB 7A40  NXTKEY GOSUB  CLCKEY        WAIT FOR KEY, FIND KEY DEFINITION
 885 158AF 55F          GONC   DSPLAY        IF ILLEGAL KEY, REPEAT DSPLAY
 886 158B2 75FD =NXTCHR GOSUB  KEYCHR        GET NEXT CHAR IN KEY DEFINITION
 887 158B6 5EE          GONC   DSPLAY        IF DEFN EXHAUSTED, REPEAT DISPLAY
 888 158B9 765E  EDITOR GOSUB  BUFFER        PLACE CHARACTER IN BUFFER
 889 158BD 54F          GONC   NXTCHR        IF BUFFER ERROR, REPEAT NXTCHR
```

```
890 158C0 771F          GOSUB  ACTIVE          ACTIVATION CHARACTER?
891 158C4 5DE           GONC   NXTCHR          IF NOT, REPEAT NXTCHR
892 158C7 7CA0          GOSUB  REDUCE          REDUCE EXPRESSION
893 158CB 7C60          GOSUB  ERRMSG          DISPLAY MESSAGE IF NECESSARY
894 158CF 59E           GONC   EDITOR          IF ARG COUNT ERROR, FORCE EDIT
895 158D2 7334          GOSUB  SYNTAX          CHECK FOR, DESTROY SYNTAX ERROR
896 158D6 5BD           GONC   NXTCHR          IF EXPR INCOMPLETE, REPEAT NXTCHR
897 158D9 7FE4          GOSUB  RESULT          STORE RESULT OR PROCESS STATEMENT
898 158DD 5BD           GONC   EDITOR          IF ILLEGAL STATEMENT, FORCE EDIT
899 158E0 8E00          GOSUBL =FINAL          DISPLAY FINAL RESULT
          00
900 158E6 789C          GOSUB  PSHSTM          DELETE OLDEST STMT, CLEAR STACKS
901 158EA 60CF          GOTO   NXTKEY          REPEAT NXTKEY
902
903 158EE 21   =CLCERR P=     1
904 158F0 0D            P=P-1                  CLEAR CARRY
905 158F2 84D           ST=0   13              CLEAR PROGRAM RUNNING FLAG
906 158F5 63CF          GOTO   EDITOR          GOTO EDITOR
907             ********************************************************************
908             ********************************************************************
909             **
910             ** Name:    CLCKEY  -  Find Key Def And ValidatE TYPE
911             **
912             ** Category:   LOCAL
913             **
914             ** Purpose:
915             **      Find key definition, validate key type
916             **
917             ** Entry:
918             **      P      =  0
919             **
920             ** Exit:
921             **      P      =  0
922             **      Sets up DEFADR.
923             **      Carry Set => Key type OK.
924             **
925             ** Calls:      KEYRD, ATNCLR, ERRMSG, CLCCFR
926             **
927             ** Uses....... Everything
928             **
929             ** Stk lvls:   xxxxxxx
930             **
931             ** History:
932             **
933             **    Date     Programmer              Modification
934             ** --------   ----------   ------------------------------------
935             **  06/13/83   SA           Added documentation
936             **
937             ********************************************************************
938             ********************************************************************
939 158F9 8E00  CLCKEY GOSUBL =KEYRD          GET KEY VECTOR
          00
940 158FF 8E00          GOSUBL =atnclr         CLEAR ATTN FLAG
          00
941 15905 80D2          P=C    2
```

```
942 15909 3FFF          LCHEX  F0111101FFFFFFFF
          FFFF
          FF10
          1111
          0F
943 1591B 20            P=     0
944 1591D 94A           ?C=0   S              ILLEGAL KEY TYPE?
945 15920 D0            GOYES  KEYERR         IF SO, GOTO KEYERR
946 15922 B46           C=C+1  S              USER-DEFINED KEY?
947 15925 400           RTNC                  IF NOT, RETURN CARRY SET
948 15928 85A           ST=1   PAREN          DENOTE STRICT PAREN MATCHING
949 1592B 02            RTNSC                 RETURN CARRY SET
950
951 1592D 3100  KEYERR  LC(2)  =eILKEY
952 15931 7600          GOSUB  ERRMSG         DISPLAY ILLEGAL KEY MESSAGE
953 15935 8C00  clccfr  GOLONG =CLCCFR        RESET DISPLAY, RETURN CARRY CLEAR
          00
954             ********************************************************************
955             ********************************************************************
956             **
957             ** Name:    ERRMSG  -  Send Error Or Warninc IN CALC MODE
958             **
959             ** Category:   LOCAL
960             **
961             ** Purpose:
962             **      Send Error or Warning Message in CALC Mode
963             **
964             ** Entry:
965             **      P      =  0
966             **      C(B)   =  Error Number (Zero causes immediate return)
967             **
968             ** Exit:
969             **      P      =  0
970             **      Carry Clear => Line Too Long or Parameter Mismatch
971             **
972             ** Calls:      STAKUP, STAKDN, MFWRN, WAITKY
973             **
974             ** Uses..... Everything
975             **
976             ** Stk lvls:   xxxxxxx
977             **
978             ** History:
979             **
980             **    Date     Programmer             Modification
981             **  --------   ----------   ---------------------------------
982             **  06/13/83   SA           Added documentation
983             **
984             ********************************************************************
985             ********************************************************************
986 1593B 96A  =ERRMSG ?C=0   B
987 1593E 00            RTNYES
988 15940 109           R1=C
989 15943 7106          GOSUB  STAKUP
990 15947 111           A=R1
991 1594A D2            C=0    A
```

```
 992 1594C AE6          C=A    B
 993 1594F 28           P=     8
 994 15951 8F00         GOSBVL =MFWRN
           000
 995 15958 7000         GOSUB  =waitky
 996 1595C 7516         GOSUB  STAKDN
 997 15960 111          A=R1
 998 15963 3100         LC(2)  =eL2LNG
 999 15967 962          ?A=C   B
1000 1596A B0           GOYES  EM10
1001 1596C 3100         LC(2)  =ePRMIS
1002 15970 966          ?A#C   B
1003 15973 00           RTNYES
1004 15975 03    EM10   RTNCC
1005               ****************************************************************
1006               ****************************************************************
1007               **
1008               ** Name:(S) REDUCE  -  Parse And Execute Partial ExpresSIONS
1009               **
1010               ** Category:   MTHUTL
1011               **
1012               ** Purpose:
1013               **      Parse and execute partial expressions in calc mode.
1014               **
1015               ** Entry:
1016               **      P      =  0
1017               **
1018               ** Exit:
1019               **      P      =  0
1020               **
1021               ** Calls:      NTOKEN, RANGE, MEMBER, PUSH, BLDCON, NRMCON,
1022               **             STAKUP, STAKDN, FNARG, ARYARG, ARGLMT, PUSH11,
1023               **             INSRTO, ORIGIN, SKPARG, PARPRP, COMPIL, ARGCNT,
1024               **             PRCDNC, CLCEXP, CLCBTS, STKBAK
1025               **
1026               ** Uses....... Everything
1027               **
1028               ** Stk lvls:   6
1029               **
1030               ** History:
1031               **
1032               **    Date     Programmer            Modification
1033               **  --------   ----------   ----------------------------------
1034               **  06/13/83   SA           Added documentation
1035               **  08/03/83   SA           Static fix to Bug 9597.
1036               **                          Packable BSS 3 created below
1037               **                          label S0-30.
1038               **
1039               ****************************************************************
1040               ****************************************************************
1041 15977 1B00 =REDUCE D0=(5) =RFNBFR        INITIALIZE
           000
1042 1597E 6E00         GOTO   START          GOTO START
1043
1044 15982 D2    HERE   C=0    A
```

```
1045 15984 02          RTNSC
1046
1047 15986 1B00  CYCLE  DO=(5) =STMTD1
           000
1048 1598D 142   START  A=DATO A          READ RFNBFR
1049 15990 1A00        DO=(4) =RAWBFR
           00
1050 15996 146         C=DATO A          READ RAWBFR
1051 15999 D5          B=C    A
1052 1599B 184         DO=DO- 5
1053 1599E 3102        LCASC  \ \
1054 159A2 6900        GOTO   BFRTST     GOTO BFRTST
1055 159A6 171   BLANK D1=D1+ 2           INCREMENT RFNBFR
1056 159A9 133         AD1EX
1057 159AC 140   BFRTST DATO=A A          UPDATE RFNBFR
1058 159AF 8B8         ?A>=B  A          RFNBFR = RAWBFR ?
1059 159B2 0D          GOYES  HERE       IF SO, RETURN CARRY SET
1060 159B4 133         AD1EX
1061 159B7 14B         A=DAT1 B          READ NEXT CHARACTER
1062 159BA 962         ?A=C   B          BLANK?
1063 159BD 9E          GOYES  BLANK      IF SO, REPEAT BLANK
1064             *     ?ST=1  BEGIN      SEARCH FOR BEGIN BASIC TOKEN?
1065             *     GOYES  STMT       IF SO, GOTO STMT
1066 159BF 8F00        GOSBVL =NTOKEN    GET FIRST TOKEN IN RAW BUFFER
           000
1067 159C6 310F        LC(2)  =tEOL
1068 159CA 966         ?A#C   B          ENDLINE TOKEN?
1069 159CD 50          GOYES  SAVRFN     IF NOT, GOTO SAVRFN
1070 159CF 171         D1=D1+ 2
1071 159D2 3400  SAVRFN LC(5)  =STMTD1
           000
1072 159D9 137         CD1EX
1073 159DC 145         DAT1=C A          SAVE RFNBFR AS MOVED BY NXTTKN
1074 159DF D2          C=0    A
1075 159E1 873         ?ST=1  NOP        FALSE LIVE CHARACTER?
1076 159E4 00          RTNYES            IF SO, RETURN CARRY SET
1077 159E6 871         ?ST=1  STRING     STRING LEXEME?
1078 159E9 44          GOYES  S0-15      IF SO, GOTO S0-15
1079 159EB 875         ?ST=1  RSTATE     STATE 1, OPERATOR EXPECTED?
1080 159EE 54          GOYES  S1-LNK     IF SO, GOTO STATE1
1081 159F0 3178  STATE0 LC(2)  =t+
1082 159F4 962         ?A=C   B          UNARY PLUS?
1083 159F7 F8          GOYES  CYCLE      IF SO, REPEAT CYCLE
1084
1085 159F9 3318  S0-10  LCHEX  8281
           28
1086 159FF 7F35        GOSUB  Rangej     UNARY MINUS OR LOGIGAL NOT?
1087 15A03 433         GOC    S0-20      IF NOT, GOTO S0-20
1088 15A06 AE8         B=A    B
1089 15A09 7294        GOSUB  C=AMS
1090 15A0D 134         DO=C
1091 15A10 161         DO=DO+ 2
1092 15A13 14A         A=DATO B          READ TOP OF PASSIVE STACK
1093 15A16 3108        LC(2)  =t^
1094 15A1A 962         ?A=C   B          INVOLUTION OPERATOR?
```

```
1095 15A1D 01          GOYES  S0-15          IF S0, GOTO S0-15
1096 15A1F AE4         A=B    B
1097 15A22 7894        GOSUB  PUSH11         PUSH NEW OPERATOR
1098 15A26 854         ST=1   DSTATE
1099 15A29 6C5F        GOTO   CYCLE          REPEAT CYCLE
1100 15A2D 3100 S0-15  LC(2)  =eILCNT        CONTEXT ERROR
1101 15A31 02          RTNSC                 RETURN CARRY SET
1102
1103 15A33 6851 S1-LNK GOTO   STATE1         SHORT JUMP LINKAGE
1104
1105 15A37 AC0  S0-20  A=0    S
1106 15A3A AC1         B=0    S
1107 15A3D 3592        LCHEX  F0F129
          1F0F
1108 15A45 25          P=     5
1109 15A47 8E00        GOSUBL =MEMBER        RIGHT PAREN, COMMA, OR ENDLINE?
          00
1110 15A4D 4F3         GOC    S0-30          IF NOT, GOTO S0-30
1111 15A50 876         ?ST=1  LOCAL          NON-NULL LOCAL EXPRESSION?
1112 15A53 62          GOYES  S0-22          IF S0, GOTO S0-22
1113 15A55 3192        LCASC  \)\
1114 15A59 966         ?A#C   B
1115 15A5C 80          GOYES  S0-21
1116 15A5E 855         ST=1   RSTATE
1117 15A61 856         ST=1   LOCAL          BUGFIX 9597 ON 830803 -SA
1118 15A64 31F7 S0-21  LC(2)  =tRES
1119 15A68 AE5         B=C    B
1120 15A6B AEC         ABEX   B
1121 15A6E 7554        GOSUB  PUSH           PUSH RESULT TOKEN
1122 15A72 AE4         A=B    B
1123 15A75 6591        GOTO   S1-00          GOTO S1-00
1124 15A79 1B00 S0-22  DO=(5) =RFNBFR
          000
1125 15A80 146         C=DATO A
1126 15A83 164         DO=DO+ 5
1127 15A86 144         DATO=C A              BACK UP RAWBFR TO RFNBFR
1128 15A89 63AF        GOTO   S0-15          GOTO S0-15
1129
1130 15A8D 78E7 S0-30  GOSUB  =BLDCON        CONSTANT? IF S0, FLOAT
1131 15A91 4C1         GOC    S0-50           ELSE GOTO S0-50
1132 15A94 70B4        GOSUB  STAKUP         MOVE OPERATOR STK TO ACTIVE POSN
1133 15A98 7317        GOSUB  NRMCON         TIDY UP CONSTANT
1134 15A9C 7AB3        GOSUB  PSHNUM         PUSH NUMBER ON OPERAND STACK
1135 15AA0 71D4        GOSUB  STAKDN         MOVE OPERTR STK TO PASSIVE POS'N
1136 15AA4 844         ST=0   DSTATE
1137           **************
1138           *       ST=1   RSTATE
1139           *       ST=1   LOCAL          BUGFIX 9597 ON 830803 -SA
1140           *       GOTO   CYCLE
1141           **************
1142 15AA7 6870        GOTO   B9597
1143 15AAB 000         CON(3) =FIXSPC
1144           **************
1145
1146 15AAE 31C7 S0-50  LC(2)  =tFN
```

```
1147 15AB2 966          ?A#C   B          USER-DEFINED FUNCTION?
1148 15AB5 51           GOYES  S0-52      IF NOT, GOTO S0-52
1149 15AB7 7535         GOSUB  FNARG      GET FUNCTION ARGUMENT COUNT
1150 15ABB 4F4          GOC    S0-72      IF FUNCTION FOUND, GOTO S0-72
1151 15ABE 3100         LC(2)  =eFNNtF    FN NOT FOUND
1152 15AC2 02           RTNSC             RETURN CARRY SET
1153 15AC4 3100 ROWRN   LC(2)  =eROWRN    OPERAND EXPECTED
1154 15AC8 02           RTNSC             RETURN CARRY SET
1155 15ACA 86B  S0-52   ?ST=0  VAR        VARIABLE?
1156 15ACD F0           GOYES  S0-60      IF NOT, GOTO S0-60
1157 15ACF 862          ?ST=0  ARG        SIMPLE VARIABLE?
1158 15AD2 74           GOYES  S0-73      IF SO, GOTO S0-73
1159 15AD4 7BC4         GOSUB  ARYARG     GET ARRAY ARGUMENT COUNT
1160 15AD8 6A50         GOTO   S0-75      GOTO S0-75
1161
1162 15ADC 3182 S0-60   LCASC  \(\
1163 15AE0 962          ?A=C   B          LEFT PARENTHESIS?
1164 15AE3 C6           GOYES  S0-76      IF SO, GOTO S0-76
1165
1166 15AE5 33A6 S0-70   LCHEX  B36A
          3B
1167 15AEB 7354         GOSUB  Rangej     FUNCTION?
1168 15AEF 44D          GOC    ROWRN      IF NOT, GOTO ROWRN
1169 15AF2 3308         LCHEX  8D80
          D8
1170 15AF8 7644         GOSUB  Rangej     ARITHMETIC OPERATOR?
1171 15AFC 57C          GONC   ROWRN      IF SO, GOTO ROWRN
1172 15AFF D9   S0-71   C=B    A
1173 15B01 134          D0=C
1174 15B04 181          D0=D0- 2
1175 15B07 70A3         GOSUB  ARGLMT     READ ARGCOUNT LIMITS
1176 15B0B 872  S0-72   ?ST=1  ARG        ARGUMENT LIST PRESENT?
1177 15B0E C1           GOYES  S0-74      IF SO, GOTO S0-74
1178 15B10 3100         LC(2)  =eMSPAR
1179 15B14 94D          ?B#0   S          ARGUMENT LIST REQUIRED?
1180 15B17 00           RTNYES            IF SO, RETURN CARRY SET
1181 15B19 7AA3 S0-73   GOSUB  PUSH       PUSH FUNCTION TOKEN ON STACK
1182 15B1D 854          ST=1   DSTATE
1183 15B20 855  B9597    ST=1   RSTATE     STATE 1, OPERATOR EXPECTED
1184 15B23 856          ST=1   LOCAL      NON-NULL LOCAL EXPRESSION
1185 15B26 6F5E         GOTO   CYCLE      REPEAT CYCLE
1186
1187 15B2A 3100 S0-74   LC(2)  =eILPAR
1188 15B2E 948          ?A=0   S          ZERO ARGUMENT FUNCTION?
1189 15B31 00           RTNYES            IF SO, RETURN CARRY SET
1190 15B33 7093 S0-75   GOSUB  PUSH       PUSH FUNCTION TOKEN
1191 15B37 1F00         D1=(5) =STMTD1
          000
1192 15B3E 147          C=DAT1 A          MOVE LEXPOINTER PAST PARENTHESIS
1193 15B41 E6           C=C+1  A
1194 15B43 E6           C=C+1  A
1195 15B45 145          DAT1=C A
1196 15B48 3182         LCASC  \(\
1197 15B4C AEA          A=C    B
1198 15B4F AC9  S0-76   C=B    S
```

```
1199 15B52 7863          GOSUB  PUSH11        PUSH LEFT PARENTHESIS
1200 15B56 3192          LCASC  \)\           LOAD RIGHT PARENTHESIS
1201 15B5A 94E           ?C#0   S
1202 15B5D D0            GOYES  S0-78
1203 15B5F 7E9C  S0-77   GOSUB  INSRTO        INSERT CHARACTER
1204 15B63 461           GOC    S0-79         IF INSERTION FAILS, GOTO S0-79
1205 15B66 31C2          LCASC  \,\           LOAD COMMA
1206 15B6A A4E   S0-78   C=C-1  S            INSERTION REQUIRED?
1207 15B6D 51F           GONC   S0-77         IF SO, REPEAT S0-77
1208 15B70 854           ST=1   DSTATE
1209 15B73 846           ST=0   LOCAL         NEW LOCAL EXPRESSION
1210 15B76 6F0E          GOTO   CYCLE         REPEAT CYCLE
1211 15B7A 189   S0-79   D0=D0- 10
1212 15B7D 17A           D1=D1+ 11
1213 15B80 147           C=DAT1 A             READ UPDATED RFNBFR
1214 15B83 144           DAT0=C A             WRITE TO RFNBFR
1215 15B86 3100          LC(2)  =eL2LNG       LINE TOO LONG
1216 15B8A 02            RTNSC                RETURN CARRY SET
1217
1218 15B8C 32A8  STATE1  LCHEX  28A
          2
1219 15B91 936           ?A#C   X             EQUALITY OPERATOR?
1220 15B94 77            GOYES  S1-00         IF NOT, GOTO S1-00
1221 15B96 877           ?ST=1  ASSIGN        ASNMNT OPERATOR ALREADY FOUND?
1222 15B99 27            GOYES  S1-00         IF SO, GOTO S1-00
1223 15B9B 1B00          D0=(5) =AVMEMS
          000
1224 15BA2 146           C=DAT0 A
1225 15BA5 135           D1=C
1226 15BA8 D0            A=0    A
1227 15BAA 15B3          A=DAT1 4
1228 15BAE 3411          LCHEX  0F011
          0F0
1229 15BB5 8A2           ?A=C   A             HAS DESTINATION BEEN EVALUATED?
1230 15BB8 B4            GOYES  NOTAOP        IF SO, GOTO NOTAOP
1231 15BBA 7EAC          GOSUB  ORIGIN
1232 15BBE 172           D1=D1+ 3
1233 15BC1 8F00          GOSBVL =NTOKEN       GET FIRST TOKEN ON LINE
          000
1234 15BC8 86B           ?ST=0  VAR           VARIABLE?
1235 15BCB 83            GOYES  NOTAOP        IF NOT, GOTO NOTAOP
1236 15BCD 862           ?ST=0  ARG           ARRAY?
1237 15BD0 60            GOYES  AOP10         IF NOT, GOTO AOP10
1238 15BD2 7764          GOSUB  SKPARG        SKIP ARGUMENT LIST
1239 15BD6 3400  AOP10   LC(5)  =RFNBFR
          000
1240 15BDD 137           CD1EX
1241 15BE0 143           A=DAT1 A
1242 15BE3 8A6           ?A#C   A             "=" AT CURRENT RFNBFR?
1243 15BE6 D1            GOYES  NOTAOP        IF NOT, GOTO NOTAOP
1244 15BE8 31D3          LCASC  \=\
1245 15BEC AEA           A=C    B
1246 15BEF 7BC2          GOSUB  PUSH11        PUSH ASSIGNMENT OPERATOR
1247 15BF3 854           ST=1   DSTATE
1248 15BF6 845           ST=0   RSTATE
```

```
1249 15BF9 846          ST=0   LOCAL
1250 15BFC 857          ST=1   ASSIGN
1251 15BFF 668D         GOTO   CYCLE          REPEAT CYCLE
1252 15C03 32A8 NOTAOP  LCHEX  28A            REGENERATE EQUALITY OPERATOR
           2
1253 15C08 ABA          A=C    X
1254 15C0B 3198 S1-00   LC(2)  =t&
1255 15C0F 962          ?A=C   B              CONCATENATOR?
1256 15C12 23           GOYES  S1-05          IF SO, GOTO S1-05
1257 15C14 301          LC(1)  =tNOT
1258 15C17 962          ?A=C   B              LOGICAL NOT?
1259 15C1A A2           GOYES  S1-05          IF SO, GOTO S1-05
1260 15C1C 310F         LC(2)  =tEOL
1261 15C20 962          ?A=C   B              END OF LINE?
1262 15C23 E4           GOYES  S1-21          IF SO, GOTO S1-21
1263 15C25 3192         LC(2)  \)\
1264 15C29 962          ?A=C   B              RIGHT PARENTHESIS?
1265 15C2C 54           GOYES  S1-21          IF SO, GOTO S1-21
1266 15C2E 3128         LC(2)  =t-
1267 15C32 966          ?A#C   B              MINUS (AS IN SUBTRACT)?
1268 15C35 91           GOYES  S1-10          IF NOT, GOTO S1-10
1269 15C37 851          ST=1   STRING         RECORD MINUS
1270 15C3A 307          LC(1)  =t+
1271 15C3D A8A          A=C    P              CREATE PLUS TOKEN
1272 15C40 6D20         GOTO   S1-20          GOTO S1-20
1273
1274 15C44 68ED S1-05   GOTO   S0-15
1275
1276 15C48 3100 R1WRN   LC(2)  =eR1WRN
1277 15C4C 02           RTNSC
1278
1279 15C4E 3308 S1-10   LCHEX  8D80
           D8
1280 15C54 7AE2         GOSUB  Rangej         ARITHMETIC OPERATOR?
1281 15C58 551          GONC   S1-20          IF SO, GOTO S1-20
1282 15C5B 311F         LC(2)  =tCOMMA
1283 15C5F 966          ?A#C   B              COMMA?
1284 15C62 6E           GOYES  R1WRN          IF NOT, GOTO R1WRN
1285 15C64 31C2         LCASC  \,\            DECOMPILE USES ASCII COMMA
1286 15C68 AEA          A=C    B
1287 15C6B 846          ST=0   LOCAL          NEW LOCAL EXPRESSION
1288 15C6E 845 S1-20    ST=0   RSTATE         STATE 0, OPERAND EXPECTED
1289 15C71 AB8 S1-21    B=A    X
1290 15C74 1B00         DO=(5) (=S-R1-2)+2
           000
1291 15C7B 14A          A=DATO B              READ CONTEXT LIMIT
1292 15C7E 968          ?A=0   B              NULL?
1293 15C81 F0           GOYES  S1-22          IF SO, GOTO S1-22
1294 15C83 78E3         GOSUB  PRCDNC         CONTEXT ERROR?
1295 15C87 501          GONC   S1-23          IF SO, RETURN CARRY SET
1296 15C8A AE0          A=0    B
1297 15C8D 148          DATO=A B              NULLIFY CONTEXT LIMIT
1298 15C90 7454 S1-22   GOSUB  PARPRP         PREPARE FOR PARSING
1299 15C94 6310         GOTO   S1-26          GOTO S1-26
1300 15C98 855 S1-23    ST=1   RSTATE
```

```
1301 15C9B 3100          LC(2)  =ePRCER        PRECEDENCE ERROR
1302 15C9F 02            RTNSC                 RETURN CARRY SET
1303 15CA1 ACB   S1-25   C=D    S
1304 15CA4 7254          GOSUB  COMPIL         COMPILE TOKEN
1305 15CA8 7B84  S1-26   GOSUB  ARGCNT         VERIFY FUNCTION ARGUMENT COUNT
1306 15CAC 3100          LC(2)  =ePRMIS
1307 15CB0 AC7           D=C    S
1308 15CB3 400           RTNC                  IF NOT OK, RETURN CARRY SET
1309 15CB6 75B3          GOSUB  PRCDNC         NEW OPERATOR OF LOWER PRECEDENCE?
1310 15CBA 46E           GOC    S1-25          IF SO, REPEAT S1-25
1311 15CBD 185           D0=D0- 6
1312 15CC0 3311          LC(4)  (tRELOP)*#100+#11
          A8
1313 15CC6 F1            BSL    A
1314 15CC8 F1            BSL    A
1315 15CCA AE5           B=C    B
1316 15CCD DD            BCEX   A
1317 15CCF 23            P=     3
1318 15CD1 915           ?B#C   WP             RELOP?
1319 15CD4 70            GOYES  S1-27          IF NOT, GOTO S1-27
1320 15CD6 180           D0=D0- 1
1321 15CD9 24            P=     4
1322 15CDB 1541  S1-27   DAT0=C WP             PUSH NEW OPERTR ON PASSIVE STACK
1323 15CDF 20            P=     0
1324 15CE1 861           ?ST=0  STRING         CHANGESIGN REQUIRED?
1325 15CE4 F0            GOYES  S1-28          IF NOT, GOTO S1-28
1326 15CE6 3311          LC(4)  (t-)*#100+#11
          28
1327 15CEC 183           D0=D0- 4
1328 15CEF 15C3          DAT0=C 4              PUSH CHANGESIGN OPERATOR
1329 15CF3 75D5  S1-28   GOSUB  CLCEXP         EXECUTE ACTIVE OPERATORS
1330 15CF7 7D06          GOSUB  CLCBTS         RESTORE STATUS BITS
1331 15CFB 161           D0=D0+ 2
1332 15CFE 7662          GOSUB  STKBAK         MOVE UNUSED OPRTRS TO PASSIVE STK
1333 15D02 854           ST=1   DSTATE
1334 15D05 608C          GOTO   CYCLE          REPEAT CYCLE
1335            ****************************************************************
1336            ****************************************************************
1337            **
1338            ** Name:    SYNTAX  -  Purge Characters In CALC Syntax ERROR
1339            **
1340            ** Category:   LOCAL
1341            **
1342            ** Purpose:
1343            **      Purge characters involved in CALC Mode syntax error.
1344            **
1345            ** Entry:
1346            **      P      =  0
1347            **
1348            ** Exit:
1349            **      P      =  0
1350            **      Carry Set => Statement complete.
1351            **
1352            ** Calls:      CLCCFR, MOVEU1
1353            **
```

```
1354                ** Uses..... A, B, C, D, P, D0, D1
1355                **
1356                ** Stk lvls:   1
1357                **
1358                ** History:
1359                **
1360                **    Date      Programmer              Modification
1361                **  --------    ----------   ----------------------------------
1362                **  06/13/83    SA           Added documentation
1363                **
1364                ********************************************************************
1365                ********************************************************************
1366 15D09 782C =SYNTAX GOSUB  clccfr
1367 15D0D 1A00        D0=(4) =RFNBFR
           00
1368 15D13 146         C=DAT0 A              READ RFNBFR
1369 15D16 164         D0=D0+ 5
1370 15D19 142         A=DAT0 A              READ RAWBFR
1371 15D1C 8A6         ?A#C   A              LEFT RAWBUFFER CONSUMED?
1372 15D1F E0          GOYES  SNTX10         IF NOT, GOTO SNTX10
1373 15D21 164         D0=D0+ 5
1374 15D24 146         C=DAT0 A              READ CLCSTK
1375 15D27 EA          A=A-C  A              RIGHT RAWBUFFER CONSUMED?
1376 15D29 CC          A=A-1  A              IF SO, SET CARRY
1377 15D2B 01          RTN                   RETURN
1378 15D2D 06    SNTX10 RSTK=C               SAVE RFNBFR
1379 15D2F 131         D1=A                  EXAMINE END OF LEFT RAWBUFFER
1380 15D32 3192        LCASC  \)\
1381 15D36 AEA         A=C    B
1382 15D39 31C2        LCASC  \,\
1383 15D3D AE5         B=C    B
1384 15D40 31D0        LCHEX  0D
1385 15D44 AE7         D=C    B
1386 15D47 1C1   SNTX15 D1=D1- 2             BACK UP
1387 15D4A 14F         C=DAT1 B              READ CHARACTER
1388 15D4D 962         ?A=C   B              RIGHT PARENTHESIS?
1389 15D50 7F          GOYES  SNTX15         IF SO, REPEAT SNTX15
1390 15D52 961         ?B=C   B              COMMA?
1391 15D55 2F          GOYES  SNTX15         IF SO, REPEAT SNTX15
1392 15D57 963         ?C=D   B              EOL?
1393 15D5A DE          GOYES  SNTX15         IF SO, REPEAT SNTX15
1394 15D5C 171         D1=D1+ 2
1395 15D5F 133         AD1EX                 LEAVE RIGHT RAWBFR IN A
1396 15D62 07          C=RSTK                RECALL RFNBFR
1397 15D64 863         ?ST=0  NOP            INCOMPLETE EXPONENT?
1398 15D67 25          GOYES  SNTX40         IF NOT, GOTO SNTX40
1399 15D69 135         D1=C
1400 15D6C 31B2        LCASC  \+\
1401 15D70 AE5         B=C    B
1402 15D73 31D2        LCASC  \-\
1403 15D77 AE7         D=C    B
1404 15D7A 171   SNTX20 D1=D1+ 2             INCREMENT TEXT POINTER
1405 15D7D 14F         C=DAT1 B              READ CHARACTER
1406 15D80 961         ?B=C   B              PLUS SIGN?
1407 15D83 51          GOYES  SNTX30         IF SO, GOTO SNTX30
```

```
1408 15D85 967          ?C#D   B              MINUS SIGN?
1409 15D88 2F           GOYES  SNTX20         IF NOT, REPEAT SNTX20
1410 15D8A 171          D1=D1+ 2              INCREMENT TEXT POINTER
1411 15D8D 14F          C=DAT1 B              READ CHARACTER
1412 15D90 967          ?C#D   B              MINUS SIGN?
1413 15D93 50           GOYES  SNTX30         IF NOT, GOTO SNTX30
1414 15D95 1C1          D1=D1- 2              DECREMENT TEXT POINTER
1415 15D98 137   SNTX30 CD1EX                 LEAVE LEFT RAWBFR IN C
1416 15D9B 144   SNTX35 DAT0=C A              UPDATE RAWBFR
1417 15D9E 135          D1=C
1418 15DA1 164          D0=D0+ 5
1419 15DA4 8E00         GOSUBL =MOVEU1        DELETE GARBAGE
          00
1420 15DAA 137          CD1EX
1421 15DAD 1F00         D1=(5) =CLCSTK
          000
1422 15DB4 145          DAT1=C A              UPDATE CLCSTK
1423 15DB7 03           RTNCC
1424 15DB9 7EDF  SNTX40 GOSUB  SNTX35
1425            ********************************************************************
1426            ********************************************************************
1427            **
1428            ** Name:    KILLKY  -  Zeros MemORY AT DEFADR
1429            **
1430            ** Category:   KEYUTL
1431            **
1432            ** Purpose:
1433            **      Blitzes length and type fields of DEFADR.
1434            **
1435            ** Entry:
1436            **      NONE
1437            **
1438            ** Exit:
1439            **      C(A)   =  0
1440            **
1441            ** Calls:
1442            **
1443            ** Uses....... C(A), D1
1444            **
1445            ** Stk lvls:   0
1446            **
1447            ** History:
1448            **
1449            **    Date     Programmer             Modification
1450            **  --------   ----------   ----------------------------------
1451            **  06/13/83   SA           Added documentation
1452            **
1453            ********************************************************************
1454            ********************************************************************
1455 15DBD 1F00 =KILLKY D1=(5) =DEFADR
          000
1456 15DC4 D2           C=0    A
1457 15DC6 1553         DAT1=C X              KILL REMAINDER OF KEY DEFINITION
1458 15DCA 03           RTNCC                 RETURN CARRY CLEAR
1459            ********************************************************************
```

```
1460            ****************************************************************
1461            **
1462            ** Name:    RESULT  -  Store Result In RESREG & DestinATION
1463            **
1464            ** Category:   LOCAL
1465            **
1466            ** Purpose:
1467            **      Store result in RESREG, compute destination address (if
1468            **      there is one), and store result there.
1469            **
1470            ** Entry:
1471            **      P      =  0
1472            **
1473            ** Exit:
1474            **      P      =  0
1475            **      Carry Clear => Illegal destination.
1476            **
1477            ** Calls:      D1MSTK, POP1N, PARPRP, ARGCNT, COMPIL, CLCEXP,
1478            **             DEST, CLCBTS, RESTOR, STORE, STAKDN, ERRMSG
1479            **
1480            ** Uses....... Everything
1481            **
1482            ** Stk lvls:   6
1483            **
1484            ** History:
1485            **
1486            **    Date     Programmer             Modification
1487            **  --------   ----------   ---------------------------------
1488            **  06/13/83   SA           Added documentation
1489            **
1490            ****************************************************************
1491            ****************************************************************
1492 15DCC 8E00  RESULT GOSUBL =D1MSTK
          00
1493            *       ?ST=1  BASIC          NON-COMPUTE STATEMENT?
1494            *       GOYES  BSCCHK         IF SO, GOTO BSCCHK
1495 15DD2 1B00        D0=(5) =RESREG
          000
1496 15DD9 8F00        GOSBVL =POP1N         POP RESULT
          000
1497 15DE0 561         GONC   RSLT10         IF REAL, GOTO RSLT10
1498 15DE3 31E0        LCHEX  0E             WRITE COMPLEX SIGNATURE
1499 15DE7 14C         DAT0=C B
1500 15DEA 161         D0=D0+ 2
1501 15DED 118         C=R0
1502 15DF0 1547        DAT0=C W              WRITE IMAGINARY PART TO RESULT
1503 15DF4 16F         D0=D0+ 16
1504 15DF7 1507  RSLT10 DAT0=A W             WRITE REAL PART TO RESULT
1505 15DFB 04          SETHEX
1506 15DFD 867   ASNCHK ?ST=0  ASSIGN         ASSIGNMENT STATEMENT?
1507 15E00 00          RTNYES                 IF NOT, RETURN CARRY SET
1508 15E02 17F         D1=D1+ 16
1509 15E05 3400        LC(5)  =MTHSTK
          000
1510 15E0C 137         CD1EX
```

```
1511 15E0F 145          DAT1=C A            DELETE MATH STACK
1512 15E12 72D2         GOSUB  PARPRP
1513 15E16 167          DO=DO+ 8            COMPLETE DESTINATION PARSE
1514 15E19 7A13         GOSUB  ARGCNT       VERIFY ARGUMENT COUNT
1515 15E1D 442          GOC    BOZO         IF ERROR, GOTO BOZO
1516 15E20 76D2         GOSUB  COMPIL       COMPILE TOKEN
1517 15E24 74A4         GOSUB  CLCEXP       COMPUTE DESTINATION ADDRESS
1518 15E28 8E00         GOSUBL =DEST        SAVE DESTINATION ADDRESS
           00
1519 15E2E 76D4         GOSUB  CLCBTS
1520 15E32 7410         GOSUB  RESTOR
1521 15E36 8E00         GOSUBL =STORE       STORE RESULT IN DESTINATION
           00
1522 15E3C 7531         GOSUB  STAKDN
1523 15E40 02           RTNSC               RETURN CARRY SET
1524
1525 15E42 3100  BOZO   LC(2)  =ePRMIS
1526 15E46 64FA         GOTO   ERRMSG       RETURN CARRY CLEAR
1527
1528            *BSCCHK GOSUB  C=AMS
1529            *       DO=C
1530            *       DO=DO+ 10
1531            *       A=DATO B            READ STATEMENT TOKEN
1532            *       LC(2)  =tRDIAN
1533            *       C=C-A  B            DISPLAY FORMAT STATEMENT?
1534            *       GOC    CLCDSP       IF SO, GOTO CLCDSP
1535            *       C=C-1  B            SET RADIANS MODE?
1536            *       LC(2)  =flRAD
1537            *       GOC    CLCRAD       IF SO, GOTO CLCRAD
1538            *       GOSBVL =SFLAGC      SET DEGREES MODE
1539            *       GOTO   RESTOR       RESTORE PREVIOUS RESULT & RETURN
1540            *CLCRAD GOSUBL =SFLAGS      SET RADIANS MODE
1541            *       GOTO   RESTOR       RESTORE PREVIOUS RESULT & RETURN
1542            *
1543            *CLCDSP DO=DO+ 4
1544            *       C=DATO B
1545            *       GOSBVL =SETFMT
1546            *       GOSBVL =SETDGT
1547            ********************************************************************
1548 15E4A 1F00  RESTOR D1=(5) =RESREG
           000
1549 15E51 8F00         GOSBVL =POP1N
           000
1550 15E58 04           SETHEX
1551            ********************************************************************
1552 15E5A 1B00  PSHNUM DO=(5) =MTHSTK
           000
1553 15E61 146          C=DATO A
1554 15E64 135          D1=C                LOCATE OPERAND STACK
1555 15E67 441          GOC    PUSH-Z
1556 15E6A 1CF          D1=D1- 16
1557 15E6D 1517         DAT1=A W            PUSH REAL NUMBER ON OPERAND STACK
1558 15E71 137   NEWMTH CD1EX
1559 15E74 144          DAT0=C A            UPDATE OPERAND STACK POINTER
1560 15E77 135          D1=C
```

```
1561 15E7A 02          RTNSC                  RETURN CARRY SET
1562 15E7C 1CF   PUSH-Z D1=D1- 16
1563 15E7F 1517         DAT1=A W              PUSH REAL PART ON OPERAND STACK
1564 15E83 110          A=R0
1565 15E86 1CF          D1=D1- 16
1566 15E89 1517         DAT1=A W              PUSH IMAG PART ON OPERAND STACK
1567 15E8D 31E0         LCHEX  0E
1568 15E91 1C1          D1=D1- 2
1569 15E94 14D          DAT1=C B              PUSH COMPLEX STACK SIGNATURE
1570 15E97 1537         A=DAT1 W
1571 15E9B 65DF         GOTO   NEWMTH         GOTO NEWMTH
1572            ********************************************************************
1573 15E9F 1B00  C=AMS  D0=(5) =AVMEMS
          000
1574 15EA6 146          C=DAT0 A
1575 15EA9 01           RTN
1576            ********************************************************************
1577 15EAB 1524  ARGLMT A=DAT0 S              READ MINARGCOUNT
1578 15EAF AC8          B=A    S
1579 15EB2 160          D0=D0+ 1
1580 15EB5 1524         A=DAT0 S              READ MAXARGCOUNT
1581 15EB9 160          D0=D0+ 1
1582 15EBC 01           RTN                   RETURN
1583            ********************************************************************
1584            *NONCMP LCHEX  D4D3
1585            *       GOSUBL =RANGE         DEGREES OR RADIANS?
1586            *       GONC   NCMP10         IF SO, GOTO NCMP10
1587            *       LCHEX  0F0C01EF
1588            *       GOSBVL =XRANGE        STD, FIX, SCI, OR ENG
1589            *       RTNC                  IF NOT, RETURN CARRY SET
1590            *NCMP10 ST=0   BEGIN
1591            *       ST=1   BASIC
1592            *       GOSUB  PUSH           PUSH TOKEN
1593            *       LCASC  \ \
1594            *       A=C    B              PUSH BLANK
1595            ********************************************************************
1596 15EBE AC0   PUSH11 A=0    S
1597 15EC1 B44          A=A+1  S
1598 15EC4 AC8          B=A    S
1599 15EC7 7030  PUSH   GOSUB  TKNLEN         DETERMINE TOKEN LENGTH
1600 15ECB 70DF         GOSUB  C=AMS          READ ADDR OF PASSIVE OPERATOR STK
1601 15ECF FA           C=-C   A
1602 15ED1 809          C+P+1                 COMPUTE NEW PASSIVE STACK POINTER
1603 15ED4 FA           C=-C   A
1604 15ED6 136          CD0EX
1605 15ED9 1501         DAT0=A WP             WRITE OPERATOR TO STACK
1606 15EDD 20           P=     0
1607 15EDF 180          D0=D0- 1
1608 15EE2 1504         DAT0=A S              WRITE MAXARGCOUNT TO STACK
1609 15EE6 ACC          ABEX   S
1610 15EE9 180          D0=D0- 1
1611 15EEC 1504         DAT0=A S              WRITE MINARGCOUNT TO STACK
1612 15EF0 ACC          ABEX   S
1613 15EF3 136          CD0EX
1614 15EF6 144          DAT0=C A              SAVE NEW PASSIVE STACK POINTER
```

```
1615 15EF9 03          RTNCC                RETURN CARRY CLEAR
1616           ********************************************************************
1617 15EFB 33C7 TKNLEN LCHEX  7D7C
           D7
1618 15F01 7D30        GOSUB  Rangej        FN OR ARRAY TOKEN?
1619 15F05 3406        LCHEX  06960
           960
1620 15F0C 4D0         GOC    TKNLN3        IF NOT, GOTO TKNLN3
1621 15F0F 23   TKNLN1 P=     3             SET FOR 2-BYTE TOKEN
1622 15F11 906         ?A#C   P             ALPHADIGIT INDICATOR PRESENT?
1623 15F14 00          RTNYES               IF NOT, RETURN
1624 15F16 25   TKNLN2 P=     5             SET FOR 3-BYTE TOKEN
1625 15F18 02          RTNSC                RETURN
1626 15F1A 7420 TKNLN3 GOSUB  Rangej        ALPHADIGIT VARIABLE?
1627 15F1E 50F         GONC   TKNLN1        IF SO, GOTO TKNLN1
1628 15F21 313B        LC(2)  =tXFN
1629 15F25 962         ?A=C   B             XROM FUNCTION?
1630 15F28 EE          GOYES  TKNLN2        IF SO, GOTO TKNLN2
1631 15F2A 31FE        LC(2)  =tXWORD
1632 15F2E 962         ?A=C   B             XROM KEYWORD?
1633 15F31 5E          GOYES  TKNLN2        IF SO, GOTO TKNLN2
1634 15F33 31A8        LC(2)  =tRELOP
1635 15F37 21          P=     1             SET FOR 1-BYTE TOKEN
1636 15F39 966         ?A#C   B             RELOP?
1637 15F3C 00          RTNYES               IF NOT, RETURN CARRY SET
1638 15F3E 22          P=     2             SET FOR 3-NIBBLE TOKEN
1639 15F40 02          RTNSC                RETURN
1640           ********************************************************************
1641 15F42 8C00 =Rangej GOLONG =RANGE
           00
1642           ********************************************************************
1643 15F48 1B00 =STAKUP DO=(5) =CLCSTK       MOVE OPERTR STK TO ACTIVE POS'N
           000
1644 15F4F 142         A=DAT0 A
1645 15F52 131         D1=A
1646 15F55 16E         DO=DO+ 15
1647 15F58 142         A=DAT0 A
1648 15F5B 164         DO=DO+ 5
1649 15F5E 8E00  STKUP1 GOSUBL =MOVEU1
           00
1650 15F64 6F20        GOTO   NEWAMS        GOTO NEWAMS
1651           ********************************************************************
1652 15F68 132  =STKBAK ADOEX               MOVE UNUSED OPRTRS TO PASSIVE STK
1653 15F6B 703F        GOSUB  C=AMS
1654 15F6F DE          ACEX   A
1655 15F71 6C10        GOTO   STKDN1        GOTO STKDN1
1656
1657 15F75 1B00 =STAKDN DO=(5) =MTHSTK       MOVE OPRTR STACK TO PASSIVE POS'N
           000
1658 15F7C 142         A=DAT0 A
1659 15F7F 131         D1=A
1660 15F82 184         DO=DO- 5
1661 15F85 142         A=DAT0 A
1662 15F88 18E         DO=DO- 15
1663 15F8B 146         C=DAT0 A
```

```
1664 15F8E 8E00  STKDN1 GOSUBL =MOVED2
          00
1665 15F94 137   NEWAMS CD1EX
1666 15F97 1F00         D1=(5) =AVMEMS        UPDATE AV MEMORY START POINTER
          000
1667 15F9E 145          DAT1=C A
1668 15FA1 03           RTNCC                 RETURN
1669                 ****************************************************************
1670 15FA3 131   ARYARG D1=A                   SAVE TOKEN IN D1
1671 15FA6 1B00         D0=(5) =FUNCRO
          000
1672 15FAD 1585         DAT0=A 6              WRITE NAME TO UNUSED MEMORY
1673 15FB1 161          D0=D0+ 2
1674 15FB4 8E00         GOSUBL =ADDRSS        DETERMINE POINTER ADDRESS
          00
1675 15FBA 133          AD1EX                 RESTORE TOKEN
1676 15FBD 442          GOC    ARY10          IF VAR NONEXISTENT, GOTO ARY10
1677 15FC0 31A1         LCHEX  1A
1678 15FC4 AE7          D=C    B
1679 15FC7 14E          C=DAT0 B              READ TYPE & DIMENSION INFORMATION
1680 15FCA 983          ?C<D   P              SIMPLE REAL VARIABLE?
1681 15FCD 51           GOYES  ARY10          IF SO, GOTO ARY10
1682 15FCF 9E3          ?C<D   B              SIMPLE NON-REAL VARIABLE?
1683 15FD2 01           GOYES  ARY10          IF SO, GOTO ARY10
1684 15FD4 BE6          CSR    B              PLACE ARGUMENT COUNT
1685 15FD7 816          CSRC
1686 15FDA ACA          A=C    S
1687 15FDD AC5          B=C    S
1688 15FE0 03           RTNCC                 RETURN CARRY CLEAR
1689 15FE2 AC0   ARY10  A=0    S              CREATE [1,2] ARGUMENT COUNT
1690 15FE5 B44          A=A+1  S
1691 15FE8 AC8          B=A    S
1692 15FEB B44          A=A+1  S
1693 15FEE 03           RTNCC                 RETURN CARRY CLEAR
1694                 ****************************************************************
1695 15FF0 1B00  FNARG  D0=(5) =FUNCRO
          000
1696 15FF7 1585         DAT0=A 6
1697 15FFB 161          D0=D0+ 2
1698 15FFE AA0          A=0    XS
1699 16001 14A          A=DAT0 B
1700 16004 3306         LCHEX  6960
          96
1701 1600A 743F         GOSUB  Rangej
1702 1600E 411          GOC    FNARG1
1703 16011 BE0          ASL    B
1704 16014 BB0          ASL    X
1705 16017 B24          A=A+1  XS
1706 1601A 161          D0=D0+ 2
1707 1601D 14A          A=DAT0 B
1708 16020 101   FNARG1 R1=A
1709 16023 8E00         GOSUBL =FNDFCN        FIND USER-DEFINED FUNCTION
          00
1710 16029 1524         A=DAT0 S
1711 1602D AC8          B=A    S
```

```
1712 16030 1B00          DO=(5) =FUNCRO
           000
1713 16037 15A5          A=DATO 6
1714 1603B 01            RTN
1715             ********************************************************************
1716 1603D 31D0  SKPARG  LCHEX  0D
1717 16041 AE5           B=C    B
1718 16044 D3            D=0    A              INITIALIZE
1719 16046 CF    SKP10   D=D-1  A              DECREMENT PARENTHESIS COUNT
1720 16048 171   SKP20   D1=D1+ 2              INCREMENT TEXT POINTER
1721 1604B 14B           A=DAT1 B              READ NEXT CHARACTER
1722 1604E 960           ?A=B   B              EOL?
1723 16051 00            RTNYES                 IF SO, RETURN CARRY SET
1724 16053 3182          LCASC  \(\
1725 16057 962           ?A=C   B              LEFT PARENTHESIS?
1726 1605A CE            GOYES  SKP10          IF SO, REPEAT SKP10
1727 1605C E6            C=C+1  A
1728 1605E 966           ?A#C   B              RIGHT PARENTHESIS?
1729 16061 7E            GOYES  SKP20          IF NOT, REPEAT SKP20
1730 16063 E7            D=D+1  A              INCREMENT PARENTHESIS COUNT
1731 16065 52E           GONC   SKP20          IF NOT OUT, REPEAT SKP20
1732 16068 8D00          GOVLNG =GNXCR+        SKIP TO NON-BLK & RETURN CRY SET
           000
1733             ********************************************************************
1734 1606F 370F  PRCDNC  LCHEX  3D2C28F0
           82C2
           D3
1735 16079 27            P=     7              SPACE, ASNMNT OPERATOR, COMMA,
1736 1607B 8E00          GOSUBL =MEMBER        LEFT PAREN, OR EOL FROM STACK?
           00
1737 16081 500           RTNNC                 IF SO, RETURN CARRY CLEAR
1738 16084 310F          LC(2)  =tEOL
1739 16088 961           ?B=C   B              NEW EOL?
1740 1608B 00            RTNYES                 IF SO, RETURN CARRY SET
1741 1608D 3192          LCASC  \)\
1742 16091 961           ?B=C   B              NEW RIGHT PARENTHESIS?
1743 16094 00            RTNYES                 IF SO, RETURN CARRY SET
1744 16096 31C2          LCASC  \,\
1745 1609A 961           ?B=C   B              NEW COMMA?
1746 1609D 00            RTNYES                 IF SO, RETURN CARRY SET
1747 1609F 3308          LCHEX  8D80
           D8
1748 160A5 799E          GOSUB  Rangej         ARITHMETIC OPERATOR?
1749 160A9 400           RTNC                  IF NOT, RETURN CARRY SET
1750 160AC AE9           C=B    B
1751 160AF 80D0          P=C    0
1752 160B3 3F77          LCHEX  6666555432211877
           8112
           2345
           5566
           66
1753 160C5 AA7           D=C    XS
1754 160C8 AE6           C=A    B
1755 160CB 80D0          P=C    0
1756 160CF 3F88          LCHEX  7777666543322988
```

```
                9223
                3456
                6677
                77
  1757 160E1 B23          D=D-C  XS
  1758 160E4 20           P=     0
  1759 160E6 01           RTN                    IF NEW OPERTR NOT HIGHER, SET CRY
  1760               ********************************************************************
  1761 160E8 73BD =PARPRP GOSUB  C=AMS           INITIALIZE FOR PARSE
  1762 160EC 18E          D0=D0- 15
  1763 160EF 142          A=DAT0 A
  1764 160F2 134          D0=C
  1765 160F5 131          D1=A
  1766 160F8 01           RTN
  1767               ********************************************************************
  1768 160FA 7DFD =COMPIL GOSUB  TKNLEN          DELETE TOKEN FROM PASSIVE STACK
  1769 160FE 136          CD0EX
  1770 16101 809          C+P+1
  1771 16104 134          D0=C
  1772 16107 1511         DAT1=A WP              APPEND TO ACTIVE STACK
  1773 1610B 137          CD1EX
  1774 1610E 809          C+P+1
  1775 16111 135          D1=C
  1776 16114 20           P=     0
  1777 16116 35D7         LCHEX  B37C7D
                C73B
  1778 1611E 25           P=     5
  1779 16120 8E00         GOSUBL =MEMBER         XFN, USER-DEF FUNCTION, OR ARRAY?
                00
  1780 16126 570          GONC   CMPL10          IF SO, GOTO CMPL10
  1781 16129 940          ?A=B   S               UNIQUE ARGUMENT COUNT?
  1782 1612C 00           RTNYES                 IF SO, RETURN CARRY SET
  1783 1612E 1554  CMPL10 DAT1=C S               APPEND PARM COUNT TO ACTIVE STACK
  1784 16132 170          D1=D1+ 1
  1785 16135 02           RTNSC                  RETURN CARRY SET
  1786               ********************************************************************
  1787 16137 AC2  =ARGCNT C=0    S               INITIALIZE ACTUAL ARGUMENT COUNT
  1788 1613A 7D6D         GOSUB  ARGLMT          POP ARGCOUNT LIMITS
  1789 1613E 15A5         A=DAT0 6               READ TOKEN
  1790 16142 3192         LCASC  \)\
  1791 16146 966          ?A#C   B               RIGHT PARENTHESIS FROM STACK?
  1792 16149 C2           GOYES  ARG30           IF NOT, RETURN CARRY CLEAR
  1793 1614B 3182         LCASC  \(\
  1794 1614F B46   ARG10  C=C+1  S               INCREMENT PARAMETER COUNT
  1795 16152 400          RTNC                   IF MORE THAN 15, RETURN CARRY SET
  1796 16155 163          D0=D0+ 4
  1797 16158 14A          A=DAT0 B               READ TOKEN FROM STACK
  1798 1615B 966          ?A#C   B               LEFT PARENTHESIS?
  1799 1615E 1F           GOYES  ARG10           IF NOT, REPEAT ARG10
  1800 16160 161          D0=D0+ 2
  1801 16163 744D         GOSUB  ARGLMT          POP ARGCOUNT LIMITS
  1802 16167 15A5         A=DAT0 6               READ TOKEN
  1803 1616B 9C2  ARG20   ?A<C   S               ARGUMENT COUNT TOO LARGE?
  1804 1616E 90           GOYES  ARG40           IF SO, GOTO ARG40
  1805 16170 9C1          ?B>C   S               ARGUMENT COUNT TOO SMALL?
```

```
1806 16173 00          RTNYES              IF SO, RETURN CARRY SET
1807 16175 03    ARG30 RTNCC               RETURN CARRY CLEAR
1808 16177 370F  ARG40 LCHEX  3D2C28F0     IMPLICIT COMPLEX DETECTION
           82C2
           D3
1809 16181 27          P=     7
1810 16183 8E00        GOSUBL =MEMBER      DELIMITER?
           00
1811 16189 5F0         GONC   ARG50        IF SO, GOTO ARG50
1812 1618C 3308        LCHEX  8D80
           D8
1813 16192 7CAD        GOSUB  Rangej       OPERATOR?
1814 16196 400         RTNC                IF NOT, RETURN CARRY SET
1815 16199 A4E   ARG50 C=C-1  S            DECREMENT ACTUAL ARGCOUNT
1816 1619C 7BCF        GOSUB  ARG20        ACTUAL ARGCOUNT OK?
1817 161A0 400         RTNC                IF NOT, RETURN CARRY SET
1818 161A3 31A7        LC(2)  =tCMPLX
1819 161A7 14D         DAT1=C B            COMPILE COMPLEX TOKEN
1820 161AA 171         D1=D1+ 2
1821 161AD 03          RTNCC               RETURN CARRY CLEAR
1822
1823          ********************************************************************
1824          ********************************************************************
1825          **
1826          ** Name:(S) NRMCON  -  Convert BLDCON Constant into Usable Form
1827          **
1828          ** Category:   MTHUTL
1829          **
1830          ** Purpose:
1831          **      Converts a 12-digit constant built by BLDCON into a
1832          **      nice normalized number taking into account overflow and
1833          **      underflow with appropriate trap settings.
1834          **
1835          ** Entry:
1836          **      Exit conditions of BLDCON.
1837          **
1838          ** Exit:
1839          **      A      =  12-digit normalized number.
1840          **      XM=0 iff number ok (no overflow or underflow)
1841          **      May generate warning message if XM=1.
1842          **
1843          ** Calls:      SFLAGS, MFWRNQ
1844          **
1845          ** Uses.......
1846          **      A-D,D0,D1,R0,P
1847          **
1848          ** Stk lvls:   3
1849          **
1850          ** History:
1851          **
1852          **    Date     Programmer          Modification
1853          **  --------   ----------   ----------------------------------
1854          **             SA           Wrote
1855          **  11/01/83   NM           Attempted to document
1856          **  12/16/83   FH           Added more documentation, changed
```

```
1857            **                          name from GRONK to NRMCON, made
1858            **                          a supported entry point
1859            **
1860            ************************************************************
1861            ************************************************************
1862 161AF AF4  =NRMCON A=B    W
1863 161B2 831         ?XM=0               BIG OR SMALL?
1864 161B5 74          GOYES  NRMCN0       IF NOT, GOTO NRMCN0
1865 161B7 1F00        D1=(5) =OVFNIB
            000
1866 161BE 3101        LC(2)  =tBIG
1867 161C2 966         ?A#C   B            BIG TOKEN?
1868 161C5 66          GOYES  NRMCN3       IF NOT, GOTO NRMCN3
1869 161C7 1574        C=DAT1 S            READ OVERFLOW NIBBLE
1870 161CB AC5         B=C    S
1871 161CE 3100        LC(2)  =eOVFLW
1872 161D2 A4D         B=B-1  S            CROAK?
1873 161D5 482         GOC    NRMCN1       IF SO, GOTO NRMCN1
1874 161D8 3100        LC(2)  =flOVF
1875 161DC 8E00        GOSUBL =SFLAGS      SET OVERFLOW FLAG
            00
1876 161E2 3300        LC(4)  =eOVFLW
            00
1877 161E8 28          P=     8
1878 161EA 949         ?B=0   S            CURSE?
1879 161ED 92          GOYES  NRMCN2       IF SO, GOTO NRMCN2
1880 161EF 8F00        GOSBVL =MFWRNQ      SWEAR AT USER ANYWAY
            000
1881 161F6 AF0         A=0    W            CONSTRUCT INFINITY
1882 161F9 A2C         A=A-1  XS
1883 161FC 03   NRMCN0 RTNCC               RETURN CARRY CLEAR
1884
1885 161FE 1E00 NRMCN1 D1=(4) =STMTD1
            00
1886 16204 143         A=DAT1 A
1887 16207 1E00        D1=(4) =RFNBFR
            00
1888 1620D 141         DAT1=A A
1889 16210 8C00 mferr  GOLONG =MfErr       BYE BYE!
            00
1890
1891 16216 8F00 NRMCN2 GOSBVL =MFWRNQ      SWEAR AT USER
            000
1892 1621D 8F00        GOSBVL =BIG+        CONSTRUCT MAXREAL
            000
1893 16224 AFA         A=C    W
1894 16227 04          SETHEX
1895 16229 03          RTNCC               RETURN CARRY CLEAR
1896
1897 1622B 1C0  NRMCN3 D1=D1- 1
1898 1622E 1574        C=DAT1 S            READ UNDERFLOW NIBBLE
1899 16232 AC5         B=C    S
1900 16235 3100        LC(2)  =eUNFLW
1901 16239 A4D         B=B-1  S            CROAK?
1902 1623C 41C         GOC    NRMCN1       IF SO, GOTO NRMCN1
```

```
1903 1623F 101          R1=A
1904 16242 3100         LC(2)  =flUNF
1905 16246 8E00         GOSUBL =SFLAGS        SET UNDERFLOW FLAG
           00
1906 1624C 3300         LC(4)  =eUNFLW
           00
1907 16252 28           P=     8
1908 16254 949          ?B=0   S              CURSE?
1909 16257 61           GOYES  NRMCN4         IF SO, GOTO NRMCN4
1910 16259 8F00         GOSBVL =MFWRNQ        SWEAR AT USER ANYWAY
           000
1911 16260 111          A=R1                  RETURN DENORMALIZED NUMBER
1912 16263 3210         LCHEX  501
           5
1913 16268 ABA          A=C    X
1914 1626B 03           RTNCC                 RETURN CARRY CLEAR
1915
1916 1626D 8F00  NRMCN4 GOSBVL =MFWRNQ        SWEAR AT USER
           000
1917 16274 AF0          A=0    W              RETURN ZERO
1918 16277 03           RTNCC                 RETURN CARRY CLEAR
1919            ********************************************************************
1920            * Output is in B-reg., B(S) does not indicate the sign of the
1921            * number.
1922            * XM = 0 at exit if the number is not over/under flow.
1923            *
1924            ********************************************************************
1925            ********************************************************************
1926            **
1927            ** Name:(S) BLDCON  -  Build A Constant For CalC MODE
1928            **
1929            ** Category:   MTHUTL
1930            **
1931            ** Purpose:
1932            **      Build a constant for calc mode.
1933            **
1934            ** Entry:
1935            **      Exit conditions of NUMSCN.
1936            **
1937            ** Exit:
1938            **    If XM = 0:  (no Overflow or Underflow)
1939            **      B      =  Normalized unsigned 12-digit number.
1940            **    If XM = 1:  (Overflow or Underflow occurred)
1941            **      B(B)   =  Token indicating overflow (=tBIG) or
1942            **                underflow (=tSMALL).
1943            **
1944            ** Calls:      None.
1945            **
1946            ** Uses.......
1947            **             A,B,C, XM.
1948            **
1949            ** Stk lvls:   1
1950            **
1951            ** History:
1952            **
```

```
1953                **    Date     Programmer          Modification
1954                **  --------   ----------  ---------------------------------
1955                **             SA          Wrote
1956                **  11/01/83   NM          Attempted to document
1957                **  12/16/83   FH          Added more documentation
1958                **
1959                ****************************************************************
1960                ****************************************************************
1961 16279 821   =BLDCON XM=0
1962 1627C 8E00         GOSUBL =DRANGE        ONE-DIGIT CONSTANT?
          00
1963 16282 451          GOC    BC20           IF NOT, GOTO BC20
1964 16285 AF1          B=0    W              CONSTRUCT FLOATINGPOINT NUMBER
1965 16288 A88          B=A    P
1966 1628B 815          BSRC
1967 1628E BF5          BSR    W
1968 16291 03           RTNCC                 RETURN CARRY CLEAR
1969
1970 16293 AB8   BC10   B=A    X              SAVE PRIMARY TOKEN
1971 16296 00           RTNSXM                RETURN CARRY CLEAR, XM SET
1972 16298 33    BC20   NIBHEX 33
1973 1629A 11           CON(2) =tSMALL
1974 1629C 01           CON(2) =tBIG
1975 1629E 23           P=     3
1976 162A0 8E00         GOSUBL =MEMBER        BIG OR SMALL?
          00
1977 162A6 5CE          GONC   BC10           IF SO, GOTO BC10
1978 162A9 968          ?A=0   B
1979 162AC 00           RTNYES
1980 162AE 31E1         LCHEX  1E
1981 162B2 9EE          ?A>=C  B              MULTIDIGIT CONSTANT?
1982 162B5 00           RTNYES                IF NOT, RETURN CARRY SET
1983 162B7 A0A          A=A+C  P              FORTUITOUS EVENT
1984 162BA 6600         GOTO   BC40           GOTO BC40
1985 162BE BD1   BC30   BSL    M              NORMALIZE MANTISSA
1986 162C1 A0C   BC40   A=A-1  P
1987 162C4 59F          GONC   BC30
1988 162C7 AC1          B=0    S              CLEAR SIGN
1989 162CA 03           RTNCC                 RETURN CARRY CLEAR
1990                ****************************************************************
1991 162CC 310F  =CLCEXP LC(2)  =tEOL
1992 162D0 14D          DAT1=C B              APPEND EXECUTION TERMINATOR
1993 162D3 171          D1=D1+ 2
1994 162D6 132          ADOEX
1995 162D9 1B00         D0=(5) =MTHSTK
          000
1996 162E0 7A7C         GOSUB  STKUP1         MOVE PASSIVE OPERATORS
1997 162E4 1C4          D1=D1- 5
1998 162E7 145          DAT1=C A              INITIALIZE UNUSED POINTERS
1999 162EA 1C4          D1=D1- 5
2000 162ED 145          DAT1=C A
2001 162F0 1C4          D1=D1- 5
2002 162F3 143          A=DAT1 A
2003 162F6 132          ADOEX                 INSTALL PC
2004 162F9 8E00         GOSUBL =GETST         GET EXPRESSION STATUS BITS
```

```
            00
2005 162FF 131          D1=A                INSTALL SP
2006 16302 8C00  =Expr  GOLONG =EXPR        EXECUTE EXPRESSION & RETURN
            00
2007            ********************************************************************
2008 16308 30A  =CLCBTS LCHEX  A
2009 1630B 982          ?A<C   P            REAL NUMBER ON STACK?
2010 1630E B0           GOYES  CBITS1       IF SO, GOTO CBITS1
2011 16310 31E0         LCHEX  0E
2012 16314 966          ?A#C   B            COMPLEX NUMBER?
2013 16317 F0           GOYES  CBITS2       IF NOT, GOTO CBITS2
2014 16319 8F00  CBITS1 GOSBVL =NOPRGM      TURN OFF PROGRAM ANNUNCIATOR
            000
2015 16320 8C00 =rstst  GOLONG =RSTST       RESTORE STATUS BITS
            00
2016
2017 16326 8C00  CBITS2 GOLONG =RDATTY
            00
2018
```

```
 ACTIVE  Abs   88027 #157DB -   714   890
 ACTV10  Abs   88063 #157FF -   726   722
 ADDRSS  Ext               -  1674
 AOP10   Abs   89046 #15BD6 -  1239  1237
 ARG     Abs       2 #00002 -   255  1157  1176  1236
 ARG10   Abs   90447 #1614F -  1794  1799
 ARG20   Abs   90475 #1616B -  1803  1816
 ARG30   Abs   90485 #16175 -  1807  1792
 ARG40   Abs   90487 #16177 -  1808  1804
 ARG50   Abs   90521 #16199 -  1815  1811
=ARGCNT  Abs   90423 #16137 -  1787  1305  1514
 ARGLMT  Abs   89771 #15EAB -  1577  1175  1788  1801
 ARY10   Abs   90082 #15FE2 -  1689  1676  1681  1683
 ARYARG  Abs   90019 #15FA3 -  1670  1159
 ASNCHK  Abs   89597 #15DFD -  1506
 ASSIGN  Abs       7 #00007 -   260  1221  1250  1506
 AVMEMS  Ext               -   775  1223  1573  1666
 B9597   Abs   88864 #15B20 -  1183  1142
 BACKUP  Abs   88187 #1587B -   841   845
 BASIC   Abs       8 #00008 -   261
 BC10    Abs   90771 #16293 -  1970  1977
 BC20    Abs   90776 #16298 -  1972  1963
 BC30    Abs   90814 #162BE -  1985  1987
 BC40    Abs   90817 #162C1 -  1986  1984
 BEGIN   Abs       0 #00000 -   253
 BFR30   Abs   87903 #1575F -   641   635   660
 BFR40   Abs   87946 #1578A -   656   633
 BFR50   Abs   87969 #157A1 -   664   658
 BFR51   Abs   88008 #157C8 -   672   668
 BFR52   Abs   88015 #157CF -   674   671
 BFR55   Abs   87914 #1576A -   646   637   640   663
 BFR60   Abs   87916 #1576C -   648   675
 BFRTST  Abs   88492 #159AC -  1057  1054
 BIG+    Ext               -  1892
 BLANK   Abs   88486 #159A6 -  1055  1063
=BLDCON  Abs   90745 #16279 -  1961  1130
 BOZO    Abs   89666 #15E42 -  1525  1515
 BUFFER  Abs   87827 #15713 -   618   888
 BUILD   Ext               -   883
 C=AMS   Abs   89759 #15E9F -  1573  1089  1600  1653  1761
=CALC    Abs   88206 #1588E -   878
 CBITS1  Abs   90905 #16319 -  2014  2010
 CBITS2  Abs   90918 #16326 -  2017  2013
 CLCBFR  Ext               -   299
=CLCBTS  Abs   90888 #16308 -  2008  1330  1519
 CLCCFR  Ext               -   953
=CLCERR  Abs   88302 #158EE -   903
 CLCET1  Ext               -   653
=CLCEXP  Abs   90828 #162CC -  1991  1329  1517
 CLCKEY  Abs   88313 #158F9 -   939   884
 CLCSTK  Ext               -  1421  1643
 CLRXDS  Ext               -   344
 CMPL10  Abs   90414 #1612E -  1783  1780
=CNTCHR  Abs   88182 #15876 -   839
=COMPIL  Abs   90362 #160FA -  1768  1304  1516
```

```
 CRLFOF  Ext                    -   343
 CTRL    Abs   87933 #1577D     -   653   621
 CYCLE   Abs   88454 #15986     -  1047  1083  1099  1185  1210  1251  1334
 D1MSTK  Ext                    -  1492
 DEFADR  Ext                    -   543  1455
 DEST    Ext                    -  1518
 DRANGE  Ext                    -  1962
 DSPLAY  Abs   88229 #158A5     -   883   885   887
 DSPRST  Ext                    -   345
 DSTATE  Abs       4 #00004     -   257   414  1098  1136  1182  1208  1247  1333
 EDITOR  Abs   88249 #158B9     -   888   882   894   898   906
 EM10    Abs   88437 #15975     -  1004  1000
 END10   Abs   87801 #156F9     -   568   576
 END20   Abs   87804 #156FC     -   569   566
 END30   Abs   87811 #15703     -   573   562
 END40   Abs   87825 #15711     -   578   574
 ENDDEF  Abs   87768 #156D8     -   558   546
=ERRMSG  Abs   88379 #1593B     -   986   649   893   952  1526
 EXPR    Ext                    -  2006
=Expr    Abs   90882 #16302     -  2006
 FINAL   Ext                    -   899
 FIXSPC  Ext                    -  1143
 FNARG   Abs   90096 #15FF0     -  1695  1149
 FNARG1  Abs   90144 #16020     -  1708  1702
 FNDFCN  Ext                    -  1709
 FORCE   Abs   87924 #15774     -   650   618
 FORSTK  Ext                    -   401
 FUNCRO  Ext                    -  1671  1695  1712
 GETST   Ext                    -  2004
 GNXCR+  Ext                    -  1732
 HASH2   Ext                    -   654
 HERE    Abs   88450 #15982     -  1044  1059
 INSRT0  Abs   88065 #15801     -   765  1203
 INSRT1  Abs   88068 #15804     -   766   674
 KEYCHR  Abs   87723 #156AB     -   543   886
 KEYERR  Abs   88365 #1592D     -   951   945
 KEYRD   Ext                    -   939
=KILLKY  Abs   89533 #15DBD     -  1455   342
 LOCAL   Abs       6 #00006     -   259  1111  1117  1184  1209  1249  1287
 MAXCMD  Ext                    -   297   446
 MEMBER  Ext                    -   667  1109  1736  1779  1810  1976
 MFWRN   Ext                    -   994
 MFWRNQ  Ext                    -  1880  1891  1910  1916
 MOVED1  Ext                    -   791
 MOVED2  Ext                    -  1664
 MOVEU1  Ext                    -  1419  1649
 MOVEU3  Ext                    -   507
 MTHSTK  Ext                    -  1509  1552  1657  1995
 MfErr   Ext                    -  1889
=Moveu3  Abs   87717 #156A5     -   507
 NEWAMS  Abs   90004 #15F94     -  1665  1650
 NEWMTH  Abs   89713 #15E71     -  1558  1571
=NEWSTM  Abs   87430 #15586     -   386   351
 NOP     Abs       3 #00003     -   256  1075  1397
 NOPRGM  Ext                    -  2014
```

```
 NOTAOP  Abs   89091 #15C03 -  1252  1230  1235  1243
 NRMCN0  Abs   90620 #161FC -  1883  1864
 NRMCN1  Abs   90622 #161FE -  1885  1873  1902
 NRMCN2  Abs   90646 #16216 -  1891  1879
 NRMCN3  Abs   90667 #1622B -  1897  1868
 NRMCN4  Abs   90733 #1626D -  1916  1909
=NRMCON  Abs   90543 #161AF -  1862  1133
 NTOKEN  Ext                 -  1066  1233
=NXTCHR  Abs   88242 #158B2 -   886   889   891   896
 NXTKEY  Abs   88235 #158AB -   884   901
 ORGN10  Abs   88179 #15873 -   838   450
=ORIGIN  Abs   88172 #1586C -   837  1231
 OVFNIB  Ext                 -  1865
 PAREN   Abs      10 #0000A -   263   558   636   948
=PARPRP  Abs   90344 #160E8 -  1761  1298  1512
 POP1N   Ext                 -  1496  1549
 PRCDNC  Abs   90223 #1606F -  1734  1294  1309
 PS05    Abs   87608 #15638 -   469   467
 PS10    Abs   87617 #15641 -   472   496
 PS20    Abs   87662 #1566E -   487   482
 PS30    Abs   87676 #1567C -   492   475
 PS40    Abs   87690 #1568A -   497   480
 PSHNUM  Abs   89690 #15E5A -  1552  1134
=PSHSTM  Abs   87426 #15582 -   385   900
 PUSH    Abs   89799 #15EC7 -  1599  1121  1181  1190
 PUSH-Z  Abs   89724 #15E7C -  1562  1555
 PUSH11  Abs   89790 #15EBE -  1596  1097  1199  1246
 ROWRN   Abs   88772 #15AC4 -  1153  1168  1171
 R1WRN   Abs   89160 #15C48 -  1276  1284
 RANGE   Ext                 -  1641
 RAWBFR  Ext                 -   449   627   795  1049
 RDATTY  Ext                 -  2017
=REDUCE  Abs   88439 #15977 -  1041   892
 RESREG  Ext                 -  1495  1548
 RESTOR  Abs   89674 #15E4A -  1548  1520
 RESULT  Abs   89548 #15DCC -  1492   897
 RFNBFR  Ext                 -   393   837  1041  1124  1239  1367  1887
 RSLT10  Abs   89591 #15DF7 -  1504  1497
 RSTATE  Abs       5 #00005 -   258  1079  1116  1183  1248  1288  1300
 RSTST   Ext                 -  2015
=Rangej  Abs   89922 #15F42 -  1641  1086  1167  1170  1280  1618  1626  1701
                            -  1748  1813
 S-R1-2  Ext                 -   387   624   768  1290
 S0-10   Abs   88569 #159F9 -  1085
 S0-15   Abs   88621 #15A2D -  1100  1078  1095  1128  1274
 S0-20   Abs   88631 #15A37 -  1105  1087
 S0-21   Abs   88676 #15A64 -  1118  1115
 S0-22   Abs   88697 #15A79 -  1124  1112
 S0-30   Abs   88717 #15A8D -  1130  1110
 S0-50   Abs   88750 #15AAE -  1146  1131
 S0-52   Abs   88778 #15ACA -  1155  1148
 S0-60   Abs   88796 #15ADC -  1162  1156
 S0-70   Abs   88805 #15AE5 -  1166
 S0-71   Abs   88831 #15AFF -  1172
 S0-72   Abs   88843 #15B0B -  1176  1150
```

```
 S0-73   Abs   88857 #15B19 -  1181  1158
 S0-74   Abs   88874 #15B2A -  1187  1177
 S0-75   Abs   88883 #15B33 -  1190  1160
 S0-76   Abs   88911 #15B4F -  1198  1164
 S0-77   Abs   88927 #15B5F -  1203  1207
 S0-78   Abs   88938 #15B6A -  1206  1202
 S0-79   Abs   88954 #15B7A -  1211  1204
 S1-00   Abs   89099 #15C0B -  1254  1123  1220  1222
 S1-05   Abs   89156 #15C44 -  1274  1256  1259
 S1-10   Abs   89166 #15C4E -  1279  1268
 S1-20   Abs   89198 #15C6E -  1288  1272  1281
 S1-21   Abs   89201 #15C71 -  1289  1262  1265
 S1-22   Abs   89232 #15C90 -  1298  1293
 S1-23   Abs   89240 #15C98 -  1300  1295
 S1-25   Abs   89249 #15CA1 -  1303  1310
 S1-26   Abs   89256 #15CA8 -  1305  1299
 S1-27   Abs   89307 #15CDB -  1322  1319
 S1-28   Abs   89331 #15CF3 -  1329  1325
 S1-LNK  Abs   88627 #15A33 -  1103  1080
 SAVRFN  Abs   88530 #159D2 -  1071  1069
 SE10    Abs   87349 #15535 -   302   308
 SFLAGS  Ext               -   347  1875  1905
 SKP10   Abs   90182 #16046 -  1719  1726
 SKP20   Abs   90184 #16048 -  1720  1729  1731
 SKPARG  Abs   90173 #1603D -  1716  1238
 SNAG    Abs       9 #00009 -   262
 SNTX10  Abs   89389 #15D2D -  1378  1372
 SNTX15  Abs   89415 #15D47 -  1386  1389  1391  1393
 SNTX20  Abs   89466 #15D7A -  1404  1409
 SNTX30  Abs   89496 #15D98 -  1415  1407  1413
 SNTX35  Abs   89499 #15D9B -  1416  1424
 SNTX40  Abs   89529 #15DB9 -  1424  1398
=STAKDN  Abs   89973 #15F75 -  1657   996  1135  1522
=STAKUP  Abs   89928 #15F48 -  1643   989  1132
 START   Abs   88461 #1598D -  1048  1042
 STATE0  Abs   88560 #159F0 -  1081
 STATE1  Abs   88972 #15B8C -  1218  1103
=STKBAK  Abs   89960 #15F68 -  1652  1332
=STKCMD  Abs   87533 #155ED -   446   385
 STKDN1  Abs   89998 #15F8E -  1664  1655
=STKEND  Abs   87327 #1551F -   297   349
 STKUP1  Abs   89950 #15F5E -  1649  1996
 STMTD1  Ext               -  1047  1071  1191  1885
 STORE   Ext               -  1521
 STREQL  Ext               -   478
 STRING  Abs       1 #00001 -   254  1077  1269  1324
=SYNTAX  Abs   89353 #15D09 -  1366   895
=SYSTEM  Abs   87372 #1554C -   341   878
 TKNLEN  Abs   89851 #15EFB -  1617  1599  1768
 TKNLN1  Abs   89871 #15F0F -  1621  1627
 TKNLN2  Abs   89878 #15F16 -  1624  1630  1633
 TKNLN3  Abs   89882 #15F1A -  1626  1620
 TNOFF?  Ext               -   879
 VAR     Abs      11 #0000B -   264  1155  1234
 atnclr  Ext               -   940
```

```
 clccfr  Abs   88373 #15935 -   953   348   622  1366
 eFNNtF  Ext                -  1151
 eILCNT  Ext                -  1100
 eILKEY  Ext                -   951
 eILPAR  Ext                -  1187
 eL2LNG  Ext                -   648   998  1215
 eMSPAR  Ext                -  1178
 eOVFLW  Ext                -  1871  1876
 ePRCER  Ext                -  1301
 ePRMIS  Ext                -  1001  1306  1525
 eROWRN  Ext                -  1153
 eR1WRN  Ext                -  1276
 eUNFLW  Ext                -  1900  1906
 flCALC  Ext                -   346
 flOVF   Ext                -  1874
 flUNF   Ext                -  1904
 kcBOT   Ext                -   650
 kcOFF   Ext                -   880
 mferr   Abs   90640 #16210 -  1889
=rstst   Abs   90912 #16320 -  2015
 t&      Abs     137 #00089 -    12  1254
 t+      Abs     135 #00087 -    12  1081  1270
 t-      Abs     130 #00082 -    12  1266  1326
 tBIG    Abs      16 #00010 -    12  1866  1974
 tCMPLX  Abs     122 #0007A -    12  1818
 tCOMMA  Abs     241 #000F1 -    12  1282
 tEOL    Abs     240 #000F0 -    12   407  1067  1260  1738  1991
 tFN     Abs     124 #0007C -    12  1146
 tNOT    Abs     129 #00081 -    12  1257
 tRELOP  Abs     138 #0008A -    12  1312  1634
 tRES    Abs     127 #0007F -    12  1118
 tSMALL  Abs      17 #00011 -    12  1973
 tXFN    Abs     179 #000B3 -    12  1628
 tXWORD  Abs     239 #000EF -    12  1631
 t^      Abs     128 #00080 -    12  1093
 waitky  Ext                -   995
```

Input Parameters

  Source file name is AB&CLC::MS

  Listing file name is AB/CLC:TI:ML::-1

  Object file name is AB%CLC:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                *      A    BBBB    &     BBBB   L      DDDD
 2                *     A A   B   B  & &    B   B  L       D  D
 3                *    A   A  B   B  & &    B   B  L       D  D
 4                *    A   A  BBBB    &     BBBB   L       D  D
 5                *    AAAAA  B   B  & & &  B   B  L       D  D
 6                *    A   A  B   B  &  &   B   B  L       D  D
 7                *    A   A  BBBB    && &  BBBB   LLLLL  DDDD
 8
 9                       TITLE  CALC Mode Decompiler <831212.1509>
10 1632C                ABS    #1632C
11
12                **     STATUS BIT ASSIGNMENTS
13                 BEGIN  EQU    0              BEGIN BASIC TOKEN
14                 STRING EQU    1              STRING LEXEME
15                 ARG    EQU    2              ARGUMENTS PRESENT
16                 NOP    EQU    3              FALSE REDUCTION CALL
17                 DSTATE EQU    4              DECOMPILE STATE
18                 RSTATE EQU    5              REDUCTION STATE
19                 LOCAL  EQU    6              LOCAL EXPRESSION STATE
20                 ASSIGN EQU    7              ASSIGNMENT STATEMENT
21                 BASIC  EQU    8              BASIC STATEMENT
22                 SNAG   EQU    9              EOL IS NOP
23                 SCR    EQU    10             SCRATCH
24                 VAR    EQU    11             VARIABLE
25                ****************************************************************
26 1632C 8F00 =FINAL  GOSBVL =DSPRST
         000
27 16333 7000         GOSUB  =STAKUP
28 16337 8E00         GOSUBL =D1MSTK
         00
29 1633D 8E00         GOSUBL =STR$SB
         00
30 16343 8F00         GOSBVL =REVPOP
         000
31 1634A 81C          ASRB
32 1634D AD0          A=0    M
33 16350 8F00         GOSBVL =DSPCNA
         000
34 16357 8F00         GOSBVL =CRLFOF
         000
35 1635E 8F00         GOSBVL =SCRLLR
         000
36 16365 6000         GOTO   =STAKDN
37                ****************************************************************
38                *FINAL  GOSUB  BLDPRP
39                *       DO=(4) (=DSPSTA)+3
40                *       LCHEX  5F4A00C0
41                *       DATO=C &              TURN CURSOR OFF
42                *       DO=(4) =DSPBFE
43                *       LCASC  \ \
44                *       GOSUB  DSPCHR
45                *       GOSUB  DSPNUM         DISPLAY RESULT
46                *       ADOEX
47                *       GOTO   FILL
48                ****************************************************************
```

```
 49 16369 1B00  BLDPRP DO=(5) =AVMEMS       SAVE AVMEMS POINTER
          000
 50 16370 142          A=DATO A
 51 16373 18E          DO=DO- 15
 52 16376 146          C=DATO A
 53 16379 135          D1=C
 54 1637C 141          DAT1=A A
 55 1637F 01           RTN
 56                    ************************************************************
 57 16381 74EF =BUILD  GOSUB  BLDPRP        SAVE AVMEMS POINTER
 58 16385 184          DO=DO- 5
 59 16388 146          C=DATO A
 60 1638B AF1          B=0    W
 61 1638E D5           B=C    A
 62 16390 135          D1=C
 63 16393 184          DO=DO- 5
 64 16396 146          C=DATO A
 65 16399 E1           B=B-C  A             COMPUTE LENGTH OF LEFT RAWBUFFER
 66 1639B 81D          BSRB
 67 1639E 31D0         LCHEX  0D
 68 163A2 AEA          A=C    B
 69 163A5 14F          C=DAT1 B
 70 163A8 1B00  BLD05  DO=(5) =DSPBFE
          000
 71 163AF 966          ?A#C   B             DISPLAY FIRST CHARACTER OF RIGHT
 72 163B2 01           GOYES  BLD20          RAWBUFFER
 73 163B4 3102         LCASC  \ \
 74 163B8 6900         GOTO   BLD20
 75 163BC 1C1  BLD10   D1=D1- 2
 76 163BF 14F          C=DAT1 B
 77 163C2 7E42 BLD20   GOSUB  DSPCHR        DISPLAY LEFT RAWBUFFER
 78 163C6 A6D          B=B-1  B
 79 163C9 52F          GONC   BLD10
 80 163CC 874          ?ST=1  DSTATE        DECOMPILE OPERATOR FIRST?
 81 163CF 71           GOYES  BLD40         IF SO, GOTO BLD40
 82 163D1 6010         GOTO   BLD30         GOTO BLD30
 83 163D5 3400 BLD25   LC(5)  =AVMEMS
          000
 84 163DC 137          CD1EX
 85 163DF 145          DAT1=C A
 86 163E2 7571 BLD30   GOSUB  DSPNUM        DISPLAY OPERAND
 87 163E6 7682 BLD40   GOSUB  RTNSXM
 88 163EA 1F00         D1=(5) =AVMEMS
          000
 89 163F1 147          C=DAT1 A
 90 163F4 135          D1=C
 91 163F7 7A90         GOSUB  DSPOP         DISPLAY OPERATOR
 92 163FB 49D          GOC    BLD25         IF LINE NOT COMPLETE, REPEAT BLD25
 93 163FE 3400         LC(5)  =MTHSTK
          000
 94 16405 137          CD1EX
 95 16408 145          DAT1=C A             RESTORE MTHSTK POINTER
 96 1640B 1D00         D1=(2) =CLCSTK
 97 1640F 143          A=DAT1 A
 98 16412 132          ADOEX
```

```
 99 16415 146          C=DAT0 A
100 16418 17E          D1=D1+ 15
101 1641B 145          DAT1=C A              RESTORE AVMEMS POINTER
102 1641E 3400 FILL    LC(5)  =DSPBFS
          000
103 16425 135          D1=C
104 16428 EA           A=A-C  A
105 1642A 8F00         GOSBVL =BLANKC
          000
106 16431 AFE          ACEX   W
107 16434 8C00 =stuff  GOLONG =STUFF         BLANK FILL
          00
108                    ********************************************************************
109 1643A 03   THERE   RTNCC                 RETURN CARRY CLEAR
110
111 1643C 821  TRICK1  XM=0
112 1643F 14A          A=DAT0 B
113 16442 31D2         LCASC  \-\
114 16446 962          ?A=C   B
115 16449 C4           GOYES  DSPOP
116 1644B 31B2         LCASC  \+\
117 1644F 6F30         GOTO   WRTOP1
118
119 16453 14A  TRICK2  A=DAT0 B
120 16456 31D2         LCASC  \-\
121 1645A 966          ?A#C   B
122 1645D 23           GOYES  WRTOP1
123 1645F 161          D0=D0+ 2
124 16462 6230         GOTO   DSPOP
125
126 16466 300  OPRTR   LC(1)  =t+
127 16469 902          ?A=C   P
128 1646C 0D           GOYES  TRICK1
129 1646E 300          LC(1)  =t-
130 16471 902          ?A=C   P
131 16474 FD           GOYES  TRICK2
132 16476 300          LC(1)  =tRELOP
133 16479 906          ?A#C   P
134 1647C 50           GOYES  YECCH
135 1647E 170          D1=D1+ 1
136 16481 8F00 YECCH   GOSBVL =ARITH
          000
137 16488 6800         GOTO   WRTOP2
138
139 1648C 821  WRTOP0  XM=0                  RECORD TERMINAL CONDITION
140 1648F 21   WRTOP1  P=     1
141 16491 7181 WRTOP2  GOSUB  DSPSTR         WRITE TO DISPLAY BUFFER
142 16495 831  DSPOP   ?XM=0                 FINISHED WITH OPERATORS?
143 16498 00           RTNYES                IF SO, RETURN CARRY SET
144 1649A 171          D1=D1+ 2
145 1649D 1533         A=DAT1 X
146 164A1 171          D1=D1+ 2
147 164A4 3100         LC(2)  =tEOL
148 164A8 962          ?A=C   B              FINISHED WITH LINE?
149 164AB F8           GOYES  THERE          IF SO, GOTO THERE
```

```
150 164AD 31C2          LCASC  \,\
151 164B1 962           ?A=C   B
152 164B4 8D            GOYES  WRTOPO
153 164B6 3582          LCASC   \ =(\
          D302
154 164BE 25            P=     5
155 164C0 8E00          GOSUBL =MEMBER        "(", "=", OR " "?
          00
156 164C6 58C           GONC   WRTOP1         IF SO, REPEAT WRTOP1
157 164C9 3192          LCASC  \)\
158 164CD 962           ?A=C   B              RIGHT PARENTHESIS?
159 164D0 CB            GOYES  WRTOPO         IF SO, REPEAT WRTOPO
160 164D2 3308          LCHEX  8D80
          D8
161 164D8 7000          GOSUB  =Rangej        ARITHMETIC OPERATOR?
162 164DC 598           GONC   OPRTR          IF SO, GOTO OPRTR
163 164DF 3314          LCHEX  6941
          96
164 164E5 7000          GOSUB  =Rangej        VARIABLE?
165 164E9 513           GONC   VBL            IF SO, GOTO VBL
166 164EC 3100          LC(2)  =tARRAY
167 164F0 962           ?A=C   B              ARRAY?
168 164F3 22            GOYES  ARRAY          IF SO, GOTO ARRAY
169 164F5 300           LC(1)  =tFN
170 164F8 962           ?A=C   B              USER-DEFINED FUNCTION
171 164FB 82            GOYES  FN             IF SO, GOTO FN
172 164FD 8F00          GOSBVL =GTEXT         GET TEXT FOR TOKEN
          000
173 16504 80DF          P=C    15
174 16508 119           C=R1
175 1650B 135           D1=C                  RESTORE D1 AFTER GTEXT
176 1650E AF6           C=A    W
177 16511 6F7F          GOTO   WRTOP2         REPEAT WRTOP2
178 16515 14B   ARRAY   A=DAT1 B
179 16518 171           D1=D1+ 2
180 1651B 7E10  VBL     GOSUB  VBLDC
181 1651F 617F          GOTO   WRTOP2         REPEAT WRTOP2
182 16523 14B   FN      A=DAT1 B
183 16526 171           D1=D1+ 2
184 16529 7010          GOSUB  VBLDC
185 1652D 75E0          GOSUB  DSPSTR
186 16531 3364          LCASC  \NF\
          E4
187 16537 23            P=     3
188 16539 675F          GOTO   WRTOP2         REPEAT WRTOP2
189             ****************************************************************
190 1653D 21    VBLDC   P=     1
191 1653F 306           LCHEX  6
192 16542 AEE           ACEX   B
193 16545 906           ?A#C   P              ALPHADIGIT VARIABLE?
194 16548 00            RTNYES                IF NOT, RETURN CARRY SET
195 1654A 303           LCHEX  3              CONVERT TO ASCII DIGIT
196 1654D F2            CSL    A              SHIFT LEFT
197 1654F F2            CSL    A
198 16551 14F           C=DAT1 B              READ NEXT CHARACTER
```

```
199 16554 171          D1=D1+ 2
200 16557 23           P=     3
201 16559 02           RTNSC                  RETURN CARRY SET
202            ****************************************************************
203 1655B 3400 DSPNUM  LC(5)  =MTHSTK
          000
204 16562 136          CDOEX
205 16565 D7           D=C    A
206 16567 146          C=DAT0 A
207 1656A 135          D1=C
208 1656D 8F00         GOSBVL =POP1N          POP NUMBER FROM MATH STACK
          000
209 16574 542          GONC   REAL            IF REAL, GOTO REAL
210 16577 70E0         GOSUB  SETUP           INITIALIZE POINTERS
211 1657B 120          AROEX
212 1657E 7230         GOSUB  RP-NUM          DISPLAY IMAGINARY PART
213 16582 31C2         LCASC  \,\
214 16586 7A80         GOSUB  DSPCHR          DISPLAY ","
215 1658A 120          AROEX
216 1658D 7F20         GOSUB  FMTNUM          DISPLAY REAL PART
217 16591 3182         LCASC  \(\
218 16595 6E70         GOTO   DSPCHR          DISPLAY "(" AND RETURN
219
220 16599 7EB0 REAL    GOSUB  SETUP           INITIALIZE POINTERS
221 1659D 14E          C=DAT0 B
222 165A0 AE7          D=C    B
223 165A3 31E5         LCASC  \^\
224 165A7 967          ?D#C   B               PARENTHESES POSSIBLY REQUIRED?
225 165AA 61           GOYES  FMTNUM           IF NOT, GOTO FMTNUM
226 165AC 948          ?A=0   S               PARENTHESES NECESSARY?
227 165AF 11           GOYES  FMTNUM           IF NOT, GOTO FMTNUM
228 165B1 85A          ST=1   SCR
229 165B4 D8   RP-NUM  B=A    A
230 165B6 3192         LCASC  \)\
231 165BA 7650         GOSUB  DSPCHR          DISPLAY RIGHT PARENTHESIS
232 165BE D4           A=B    A
233 165C0 1F00 FMTNUM  D1=(5) =CLCSTK
          000
234 165C7 147          C=DAT1 A
235 165CA 135          D1=C
236 165CD D7           D=C    A
237 165CF 3450         LCHEX  00005
          000
238 165D6 C3           D=C+D  A
239 165D8 147          C=DAT1 A
240 165DB 135          D1=C
241 165DE 06           RSTK=C
242 165E0 8E00         GOSUBL =CLCSTR
          00
243 165E6 07           C=RSTK
244 165E8 133          AD1EX
245 165EB 131          D1=A                   QUESTIONABLE, INVESTIGATE
246 165EE DE           ACEX   A
247 165F0 EA           A=A-C  A
248 165F2 81C          ASRB
```

```
249 165F5 AE8          B=A    B
250 165F8 A6D          B=B-1  B
251 165FB 14F   FMT20  C=DAT1 B
252 165FE 7210         GOSUB  DSPCHR
253 16602 171          D1=D1+ 2
254 16605 A6D          B=B-1  B
255 16608 52F          GONC   FMT20
256 1660B 86A          ?ST=0  SCR            LEFT PARENTHESIS REQUIRED?
257 1660E 00           RTNYES                IF NOT, RETURN
258 16610 3182         LCASC  \(\
259             ********************************************************************
260 16614 21    DSPCHR P=     1
261 16616 136   DSPSTR CDOEX
262 16619 DA           A=C    A
263 1661B D7           D=C    A
264 1661D 136          CDOEX
265 16620 1A00         DO=(4) =DSPBFS
          00
266 16626 136          CDOEX
267 16629 E3           D=D-C  A
268 1662B D2           C=0    A
269 1662D 809          C+P+1
270 16630 9E7          ?C>D   B
271 16633 F0           GOYES  DSP20
272 16635 EE    DSP10  C=A-C  A
273 16637 136          CDOEX
274 1663A 1541         DAT0=C WP
275 1663E 20           P=     0
276 16640 02           RTNSC
277 16642 DB    DSP20  C=D    A
278 16644 80D0         P=C    0
279 16648 0D           P=P-1
280 1664A 5AE          GONC   DSP10
281 1664D 130   BFRFUL DO=A
282 16650 20           P=     0
283 16652 31E7         LCASC  \~\
284 16656 14C          DAT0=C B
285 16659 02           RTNSC
286             ********************************************************************
287 1665B 04    SETUP  SETHEX
288 1665D 17F          D1=D1+ 16
289 16660 137          CD1EX
290 16663 144          DAT0=C A
291 16666 DB           C=D    A
292 16668 134          DO=C
293 1666B 84A          ST=0   SCR
294 1666E 01           RTN
295             ********************************************************************
296 16670 00    RTNSXM RTNSXM
```

ARG	Abs	2	#00002	-	15					
ARITH	Ext			-	136					
ARRAY	Abs	91413	#16515	-	178	168				
ASSIGN	Abs	7	#00007	-	20					
AVMEMS	Ext			-	49	83	88			
BASIC	Abs	8	#00008	-	21					
BEGIN	Abs	0	#00000	-	13					
BFRFUL	Abs	91725	#1664D	-	281					
BLANKC	Ext			-	105					
BLD05	Abs	91048	#163A8	-	70					
BLD10	Abs	91068	#163BC	-	75	79				
BLD20	Abs	91074	#163C2	-	77	72	74			
BLD25	Abs	91093	#163D5	-	83	92				
BLD30	Abs	91106	#163E2	-	86	82				
BLD40	Abs	91110	#163E6	-	87	81				
BLDPRP	Abs	90985	#16369	-	49	57				
=BUILD	Abs	91009	#16381	-	57					
CLCSTK	Ext			-	96	233				
CLCSTR	Ext			-	242					
CRLFOF	Ext			-	34					
D1MSTK	Ext			-	28					
DSP10	Abs	91701	#16635	-	272	280				
DSP20	Abs	91714	#16642	-	277	271				
DSPBFE	Ext			-	70					
DSPBFS	Ext			-	102	265				
DSPCHR	Abs	91668	#16614	-	260	77	214	218	231	252
DSPCNA	Ext			-	33					
DSPNUM	Abs	91483	#1655B	-	203	86				
DSPOP	Abs	91285	#16495	-	142	91	115	124		
DSPRST	Ext			-	26					
DSPSTR	Abs	91670	#16616	-	261	141	185			
DSTATE	Abs	4	#00004	-	17	80				
FILL	Abs	91166	#1641E	-	102					
=FINAL	Abs	90924	#1632C	-	26					
FMT20	Abs	91643	#165FB	-	251	255				
FMTNUM	Abs	91584	#165C0	-	233	216	225	227		
FN	Abs	91427	#16523	-	182	171				
GTEXT	Ext			-	172					
LOCAL	Abs	6	#00006	-	19					
MEMBER	Ext			-	155					
MTHSTK	Ext			-	93	203				
NOP	Abs	3	#00003	-	16					
OPRTR	Abs	91238	#16466	-	126	162				
POP1N	Ext			-	208					
REAL	Abs	91545	#16599	-	220	209				
REVPOP	Ext			-	30					
RP-NUM	Abs	91572	#165B4	-	229	212				
RSTATE	Abs	5	#00005	-	18					
RTNSXM	Abs	91760	#16670	-	296	87				
Rangej	Ext			-	161	164				
SCR	Abs	10	#0000A	-	23	228	256	293		
SCRLLR	Ext			-	35					
SETUP	Abs	91739	#1665B	-	287	210	220			
SNAG	Abs	9	#00009	-	22					
STAKDN	Ext			-	36					

```
 STAKUP  Ext                  -    27
 STR$SB  Ext                  -    29
 STRING  Abs        1 #00001  -    14
 STUFF   Ext                  -   107
 THERE   Abs    91194 #1643A  -   109   149
 TRICK1  Abs    91196 #1643C  -   111   128
 TRICK2  Abs    91219 #16453  -   119   131
 VAR     Abs       11 #0000B  -    24
 VBL     Abs    91419 #1651B  -   180   165
 VBLDC   Abs    91453 #1653D  -   190   180   184
 WRTOP0  Abs    91276 #1648C  -   139   152   159
 WRTOP1  Abs    91279 #1648F  -   140   117   122   156
 WRTOP2  Abs    91281 #16491  -   141   137   177   181   188
 YECCH   Abs    91265 #16481  -   136   134
=stuff   Abs    91188 #16434  -   107
 t+      Ext                  -   126
 t-      Ext                  -   129
 tARRAY  Ext                  -   166
 tEOL    Ext                  -   147
 tFN     Ext                  -   169
 tRELOP  Ext                  -   132
```

Input Parameters

  Source file name is AB&BLD::MS

  Listing file name is AB/BLD:TI:ML::-1

  Object file name is AB%BLD:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                *      A    BBBB    &     EEEEE  DDDD
 2                *     A A   B   B  & &    E      D  D
 3                *    A   A  B   B  & &    E      D  D
 4                *    A   A  BBBB    &     EEEE   D  D
 5                *    AAAAA  B   B  & & &  E      D  D
 6                *    A   A  B   B  &  &   E      D  D
 7                *    A   A  BBBB    && &  EEEEE  DDDD
 8
 9                       TITLE  CALC Mode Editor <831212.1510>
10 16672                ABS    #16672
11
12                      RDSYMB SB%RAM::MS
13
14                **    STATUS BIT ASSIGNMENTS
15                 BEGIN  EQU    0              BEGIN BASIC TOKEN
16                 STRING EQU    1              STRING LEXEME
17                 ARG    EQU    2              ARGUMENTS PRESENT
18                 NOP    EQU    3              FALSE REDUCTION CALL
19                 DSTATE EQU    4              DECOMPILE STATE
20                 RSTATE EQU    5              REDUCTION STATE
21                 LOCAL  EQU    6              LOCAL EXPRESSION STATE
22                 ASSIGN EQU    7              ASSIGNMENT STATEMENT
23                 BASIC  EQU    8              BASIC STATEMENT
24                 SNAG   EQU    9              EOL IS NOP
25                 PAREN  EQU    10             PARENTHESIS MATCHING
26                 VAR    EQU    11             VARIABLE
27                **
28                ****************************************************************
29                **
30 16672 0300 =CLCET1 NIBHEX 0300          -CHR - NONSENSE, RTNCC
31 16676 0300         NIBHEX 0300          lc - NONSENSE, RETURN CARRY CLEAR
32 1667A 0300         NIBHEX 0300          I/R - NONSENSE, RTNCC
33 1667E 0300         NIBHEX 0300          USER - NONSENSE
34 16682 6770         GOTO   CLCATN        -LINE - CLEAR LINE & RETURN
35 16686 6350         GOTO   CLCCFL        g< - CURSOR FAR LEFT
36 1668A 6C80         GOTO   CLCCFR        g> - CURSOR FAR RIGHT
37 1668E 6752         GOTO   CLCBKS        BKSPC - BACKSPACE
38 16692 6B90         GOTO   CLCCL         < - CURSOR LEFT
39 16696 6FA0         GOTO   CLCCR         > - CURSOR RIGHT
40 1669A 0300         NIBHEX 0300          CTRL - NONSENSE, RTNCC
41 1669E 0300         NIBHEX 0300          VIEW - NONSENSE
42 166A2 0300         NIBHEX 0300          g0 - NONSENSE
43 166A6 6381         GOTO   CLCEOL        EOL - END LINE
44 166AA 6F40         GOTO   CLCATN        ATTN - CLEAR LINE & RETURN
45 166AE 69C0         GOTO   CLCSST        RUN - PFMM OPERATION OUT OF ORDER
46 166B2 6771         GOTO   CLCEOL        CONT - END LINE
47 166B6 61C0         GOTO   CLCSST        SST - PERFORM OPERTN OUT OF ORDER
48 166BA 6B43         GOTO   CLCCB         UP - UPDATE STACK & EDIT NEW LINE
49 166BE 0300         NIBHEX 0300          DOWN - NONSENSE, RTNCC
50 166C2 6BC2         GOTO   CLCCT         TOP - UPDATE STK, EDIT OLDEST LN
51 166C6 6F33         GOTO   CLCCB         BOTTOM - UPDATE STK, EDIT NEW LN
52 166CA 6F20         GOTO   CLCATN        ATTN - CLEAR LINE & RETURN
53 166CE 6871         GOTO   CLCOUT        CALC - EXIT CALC MODE
54 166D2 6810         GOTO   CLCOFF        OFF - TURN HP-71 OFF
55 166D6 6F23         GOTO   CLCCB         gEOL - UPDATE STK, EDIT NEW LINE
```

```
     56
     57 166DA 7AA1  CLCCFL GOSUB  CHRONE
     58 166DE AE4          A=B    B
     59 166E1 1AC7         DO=(4) =FIRSTC
              4F
     60 166E7 6B60         GOTO   CRSTAT
     61
     62 166EB 7000  CLCOFF GOSUB  =STAKUP        SET MEMORY FOR SHUTDOWN
     63 166EF 8F00         GOSBVL =DSLEEP        YOU ARE GETTING SLEEPY....
              000
     64 166F6 7000         GOSUB  =STAKDN        PREPARE FOR WORK
     65 166FA 8E00  CLCATN GOSUBL =ORIGIN        TRASH YESTERDAY'S WORK
              00
     66 16700 7482         GOSUB  newstn
     67 16704 8E00         GOSUBL =atnclr        CLEAR ATTENTION FLAG
              00
     68 1670A 8F00         GOSBVL =CLRXDS        DISABLE EXERNAL DISPLAY
              000
     69 16711 8E00         GOSUBL =KILLKY        KILL REMAINDER OF KEY DEFINITION
              00
     70 16717 1B87 =CLCCFR DO=(5) (=DSPSTA)+3
              4F2
     71 1671E 3708         LCHEX  5F4A0080
              00A4
              F5
     72 16728 15C7         DATO=C 8
     73 1672C 03           RTNCC                 RETURN CARRY CLEAR
     74
     75 1672E 7651  CLCCL  GOSUB  CHRONE
     76 16732 1BC7         DO=(5) =FIRSTC
              4F2
     77 16739 14A          A=DATO B
     78 1673C A6C          A=A-1  B
     79 1673F 9E8          ?A>=B  B
     80 16742 11           GOYES  CRSTAT
     81 16744 03           RTNCC
     82
     83 16746 1BC7  CLCCR  DO=(5) =FIRSTC
              4F2
     84 1674D 14A          A=DATO B
     85 16750 B64          A=A+1  B
     86 16753 31A4  CRSTAT LCHEX  4A
     87 16757 9EE          ?A>=C  B
     88 1675A D8           GOYES  CLCCFR
     89 1675C 148          DATO=A B
     90 1675F 161          DO=DO+ 2
     91 16762 148          DATO=A B
     92 16765 185          DO=DO- 6
     93 16768 330C         LCHEX  00C0
              00
     94 1676E 15C3         DATO=C 4
     95 16772 03    RTNCC  RTNCC
     96 16774 6192  CBLNK  GOTO   CLCCB
     97
     98 16778 8E00  CLCSST GOSUBL =REDUCE        UPDATE EXPRESSION
```

```
          00
   99 1677E 8E00  SST05  GOSUBL =ERRMSG
          00
  100 16784 5FE          GONC   CBLNK          ON ERROR, FORCE EDIT
  101 16787 8E00         GOSUBL =SYNTAX        CHECK SYNTAX
          00
  102 1678D 865          ?ST=0  RSTATE         OPERATOR EXPECTED?
  103 16790 2E           GOYES  RTNCC          IF NOT, RETURN CARRY CLEAR
  104 16792 7000         GOSUB  =PARPRP        PREPARE FOR PARSING
  105 16796 3100         LC(2)  =t-
  106 1679A 6C10         GOTO   SST20          GOTO SST20
  107 1679E 161   SST10  DO=DO+ 2             DELETE TOKEN FROM PASSIVE STACK
  108 167A1 14D          DAT1=C B              APPEND TOKEN TO ACTIVE STACK
  109 167A4 171          D1=D1+ 2
  110 167A7 132          ADOEX
  111 167AA 1BD8         DO=(5) (=S-R1-2)+2
          8F2
  112 167B1 14C          DAT0=C B              RECORD TOKEN AS CONTEXT LIMIT
  113 167B4 130          DO=A
  114 167B7 161   SST20  DO=DO+ 2
  115 167BA 14A          A=DAT0 B              EXAMINE STACK
  116 167BD 962          ?A=C   B              CHANGESIGN?
  117 167C0 ED           GOYES  SST10          IF SO, REPEAT SST10
  118 167C2 181          DO=DO- 2
  119 167C5 7000         GOSUB  =ARGCNT        VERIFY ARGUMENT COUNT
  120 167C9 3100         LC(2)  =ePRMIS
  121 167CD 40B          GOC    SST05          IF ERROR, FORCE EDIT
  122 167D0 181          DO=DO- 2
  123 167D3 390F         LCHEX  203D2C28F0
          82C2
          D302
  124 167DF 29           P=     9              ASNMNT, COMMA, LEFT PARENTHESIS,
  125 167E1 8E00         GOSUBL =MEMBER        SPACE, OR ENDLINE ON STACK?
          00
  126 167E7 542          GONC   SST30          IF SO, GOTO SST30
  127 167EA 161          DO=DO+ 2
  128 167ED 7000         GOSUB  =COMPIL        COMPILE TOKEN
  129 167F1 136          CDOEX
  130 167F4 1BF8         DO=(5) (=S-R1-2)+4
          8F2
  131 167FB 1522         A=DAT0 XS
  132 167FF B24          A=A+1  XS             INCREMENT SST COUNT
  133 16802 181          DO=DO- 2
  134 16805 1503         DAT0=A X              RECORD TOKEN AS CONTEXT LIMIT
  135 16809 134          DO=C
  136 1680C 7000  SST30  GOSUB  =CLCEXP        EXECUTE FUNCTION
  137 16810 7000         GOSUB  =CLCBTS
  138 16814 161          DO=DO+ 2
  139 16817 8E00         GOSUBL =STKBAK
          00
  140 1681D 76FE         GOSUB  CLCCFR
  141 16821 844          ST=0   DSTATE         DECOMPILE OPERAND FIRST
  142 16824 8C00         GOLONG =NXTCHR        GOTO NXTCHR
          00
  143
```

```
144 1682A 79EE  CLCEOL GOSUB  CLCCFR
145 1682E 1B58         DO=(5) =CLCSTK
          5F2
146 16835 146          C=DATO A
147 16838 184          DO=DO- 5
148 1683B 144          DATO=C A
149 1683E 31D0         LCHEX  0D
150 16842 AE5          B=C    B
151 16845 02           RTNSC
152
153 16847 8E00  CLCOUT GOSUBL =ORIGIN
          00
154 1684D 133          AD1EX
155 16850 1F67         D1=(5) =CLCBFR
          5F2
156 16857 2A           P=     10
157 16859 174   CALC10 D1=D1+ 5
158 1685C 141          DAT1=A A              DESTROY CALC SYSTEM
159 1685F 0C           P=P+1
160 16861 57F          GONC   CALC10
161 16864 179          D1=D1+ 10
162 16867 143          A=DAT1 A
163 1686A 1C4          D1=D1- 5
164 1686D 141          DAT1=A A
165 16870 3100         LC(2)  =flCALC
166 16874 8E00         GOSUBL =SFLAGC
          00
167 1687A 8F00         GOSBVL =DSPRST
          000
168 16881 8D00         GOVLNG =CONFMN        JUMP TO BASIC SYSTEM
          000
169
170 16888 1BE7  CHRONE DO=(5) (=DSPBFS)-2    FIND FIRST CHARACTER IN DECOMPILE
          4F2
171 1688F AE1          B=0    B
172 16892 A6D          B=B-1  B
173 16895 3102         LCASC  \ \
174 16899 161   CHR10  DO=DO+ 2
175 1689C B65          B=B+1  B
176 1689F 14A          A=DATO B
177 168A2 962          ?A=C   B
178 168A5 4F           GOYES  CHR10
179 168A7 03           RTNCC
180
181 168A9 8E00  RENTRY GOSUBL =PSHSTM        PUSH COMMAND STACK
          00
182 168AF 8F00  ENTRY  GOSBVL =DSPRST
          000
183 168B6 1BB7         DO=(5) =RFNBFR
          5F2
184 168BD 142          A=DATO A              READ RFNBFR
185 168C0 130          DO=A
186 168C3 182          DO=DO- 3
187 168C6 1563         C=DATO X
188 168CA 93E          ?C#0   X              AT STATEMENT HEADER?
```

```
189 168CD CD          GOYES  RENTRY         IF NOT, REPEAT RENTRY
190 168CF 132         ADOEX
191 168D2 102         R2=A                  STORE COMMAND STACK POINTER
192 168D5 1A87        DO=(4) (=DSPSTA)+3
          4F
193 168DB AF2         C=0    W
194 168DE 15C7        DATO=C 8              SET DISPLAY STATUS
195 168E2 08          CLRST
196 168E4 01          RTN                   RETURN
197
198 168E6 7D2E  CLCBKS GOSUB  CLCCFR
199 168EA 1AB7        DO=(4) =RFNBFR
          5F
200 168F0 142         A=DATO A              READ RFNBFR
201 168F3 164         DO=DO+ 5
202 168F6 146         C=DATO A              READ RAWBFR
203 168F9 8A2         ?A=C   A              RFNBFR = RAWBFR?
204 168FC 53          GOYES  HRDWAY         IF SO, GOTO HRDWAY
205 168FE DA          A=C    A
206 16900 CE          C=C-1  A
207 16902 CE          C=C-1  A
208 16904 144         DATO=C A              UPDATE RAWBFR
209 16907 135         D1=C
210 1690A 164         DO=DO+ 5
211 1690D 8E00        GOSUBL =MOVEU1        DELETE CHARACTER
          00
212 16913 137         CD1EX
213 16916 1B58        DO=(5) =CLCSTK
          5F2
214 1691D 144         DATO=C A              UPDATE CLCSTK
215 16920 1AB8  DECLEN DO=(4) =S-R1-2
          8F
216 16926 14E         C=DATO B
217 16929 A6E         C=C-1  B              DECREMENT LENGTH COUNT
218 1692C 14C         DATO=C B
219 1692F 03    BSTRTN RTNCC                 RETURN CARRY CLEAR
220 16931 8E00  HRDWAY GOSUBL =CNTCHR       COUNT CHARACTERS IN BUFFER
          00
221 16937 A6F         D=D-1  B
222 1693A DB          C=D    A
223 1693C A36         C=C+C  X
224 1693F 133         AD1EX
225 16942 D8          B=A    A              SAVE ADDRESS OF START OF BUFFER
226 16944 130         DO=A
227 16947 162         DO=DO+ 3
228 1694A 1F08        D1=(5) =DSPBFS
          4F2
229 16951 8E00        GOSUBL =MOVEU3        MOVE BUFFER TO DISPLAY BUFFER
          00
230 16957 1BF8        DO=(5) (=S-R1-2)+4
          8F2
231 1695E 1564        C=DATO S              READ SST COUNT
232 16962 3100        LC(2)  =kcSST
233 16966 6C00        GOTO   SNARL          GOTO SNARL
234 1696A 14D   GROWL  DAT1=C B             APPEND SST CHAR TO DISPLAY BUFFER
```

```
    235 1696D 171          D1=D1+ 2
    236 16970 B67          D=D+1  B              INCREMENT CHARACTER COUNT
    237 16973 A4E   SNARL  C=C-1  S              DONE APPENDING SST's?
    238 16976 53F          GONC   GROWL          IF NOT, REPEAT GROWL
    239 16979 A6F          D=D-1  B              UNCOUNT LAST CHARACTER
    240 1697C 42B          GOC    BSTRTN         IF NULL BUFFER, RTNCC
    241 1697F 7960         GOSUB  PHNYKY         SET UP PHONY KEY ASSIGNMENT
    242 16983 D4           A=B    A
    243 16985 131          D1=A
    244 16988 8C00  newstm GOLONG =NEWSTM        PREPARE FOR NEW STATEMENT
              00
    245
    246 1698E 7D1F  CLCCT  GOSUB  ENTRY          INITIALIZE EDIT
    247 16992 1B67         DO=(5) =CLCBFR
              5F2
    248 16999 142          A=DAT0 A              READ CLCBFR
    249 1699C 6290         GOTO   MOVE1          GOTO MOVE1
    250
    251 169A0 112   CLCCD  A=R2                  RECALL COMMAND STACK POINTER
    252 169A3 130          DO=A
    253 169A6 D2           C=0    A
    254 169A8 1563         C=DAT0 X              READ LENGTH OF STATEMENT
    255 169AC 165          DO=DO+ 6
    256 169AF 132          ADOEX
    257 169B2 CA           A=A+C  A              COMPUTE START OF NEXT STATEMENT
    258 169B4 130          DO=A
    259 169B7 1563         C=DAT0 X              READ LENGTH OF NEXT STATEMENT
    260 169BB 93E          ?C#0   X              AT END OF COMMAND STACK?
    261 169BE A7           GOYES  MOVE2          IF NOT, GOTO MOVE2
    262 169C0 1B08         DO=(5) =DSPBFS
              4F2
    263 169C7 14C          DAT0=C B
    264 169CA 08    CLCEDX CLRST                 INITIALIZE CALC MODE STATE
    265             *      ST=1   BEGIN
    266 169CC 854          ST=1   DSTATE
    267 169CF 85A          ST=1   PAREN
    268 169D2 714D         GOSUB  CLCCFR         SET DISPLAY PARAMETERS
    269 169D6 165          DO=DO+ 6
    270 169D9 D3           D=0    A
    271 169DB B67   EXIT10 D=D+1  B              DETERMINE LEN OF DISPLAY BUFFER
    272 169DE 161          DO=DO+ 2
    273 169E1 14E          C=DAT0 B
    274 169E4 96E          ?C#0   B
    275 169E7 4F           GOYES  EXIT10
    276 169E9 A6F          D=D-1  B
    277 169EC 1B76  PHNYKY DO=(5) =DEFADR        SET UP PHONY KEY ASSIGNMENT
              9F2
    278 169F3 3700         LC(8)  (DSPBFS)*4096
              0084
              F2
    279 169FD AEB          C=D    B
    280 16A00 15C7         DAT0=C 8
    281 16A04 03           RTNCC                 RETURN CARRY CLEAR
    282
    283 16A06 75AE  CLCCB  GOSUB  ENTRY          INITIALIZE EDIT
```

```
284 16A0A 1B67  CLCCU  D0=(5) =CLCBFR
          5F2
285 16A11 142          A=DAT0 A              READ CLCBFR
286 16A14 D8           B=A    A
287 16A16 112          A=R2                 RECALL COMMAND STACK POINTER
288 16A19 8A0          ?A=B   A             POINTERS EQUAL?
289 16A1C 31           GOYES  MOVE1         IF SO, GOTO MOVE1
290 16A1E 130          D0=A
291 16A21 182          D0=D0- 3
292 16A24 D2           C=0    A
293 16A26 1563         C=DAT0 X             READ LENGTH OF PREVIOUS STATEMENT
294 16A2A 132          ADOEX
295 16A2D EA           A=A-C  A             COMPUTE START OF PREV STATEMENT
296 16A2F 130  MOVE1   D0=A
297 16A32 D2           C=0    A
298 16A34 1563         C=DAT0 X             READ LENGTH OF STATEMENT
299 16A38 162  MOVE2   D0=D0+ 3
300 16A3B 102          R2=A                 STORE COMMAND STACK POINTER
301 16A3E D7           D=C    A
302 16A40 1FC7         D1=(5) =FIRSTC
          4F2
303 16A47 D0           A=0    A
304 16A49 141          DAT1=A A             RESET CURSOR
305 16A4C 173          D1=D1+ 4
306 16A4F 8E00         GOSUBL =MOVEU3       COPY EDIT LINE TO DISPLAY BUFFER
          00
307 16A55 348D         LCHEX  000D8
          000
308 16A5C EB           C=C-D  A
309 16A5E 8E00         GOSUBL =WIPOUT       CLEAR END OF DISPLAY BUFFER
          00
310 16A64 8E00  CLCED  GOSUBL =CHEDIT
          00
311 16A6A 59F          GONC   CLCED         IGNORE IMMEDIATE EXECUTE KEY
312 16A6D 3464         LC(5)  (=CLCET2)-52
          A61
313 16A74 8C00         GOLONG =HASH2        JUMP TO NEXT FUNCTION
          00
314
315 16A7A 6F4F =CLCET2 GOTO   CLCEDX        EOL - EXECUTE EDITED TEXT & RTN
316 16A7E 6B7C         GOTO   CLCATN        ATTN - CLEAR LINE & RETURN
317 16A82 674F         GOTO   CLCEDX        RUN - EXECUTE EDITED TEXT & RTN
318 16A86 634F         GOTO   CLCEDX        CONT - EXECUTE EDITED TEXT & RTN
319 16A8A 6F3F         GOTO   CLCEDX        SST - EXECUTE EDITED TEXT & RTN
320 16A8E 6B7F         GOTO   CLCCU         UP - CURSOR UP IF POSSIBLE
321 16A92 6D0F         GOTO   CLCCD         DOWN - CURSOR DOWN IF POSSIBLE
322 16A96 67FE         GOTO   CLCCT         TOP - CURSOR TO TOP OF CMD STK
323 16A9A 6B6F         GOTO   CLCCB         BOTTOM - CURSOR TO BOT OF STACK
324 16A9E 6B5C         GOTO   CLCATN        gON - CLEAR LINE & RETURN
325 16AA2 64AD         GOTO   CLCOUT        CALC - EXIT CALC MODE
326 16AA6 644C         GOTO   CLCOFF        OFF - TURN HP-71 OFF
327 16AAA 6F4C         GOTO   CLCATN        gEOL - CLEAR LINE & RETURN
```

ARG	Abs	2	#00002	-	17						
ARGCNT	Ext			-	119						
ASSIGN	Abs	7	#00007	-	22						
BASIC	Abs	8	#00008	-	23						
BEGIN	Abs	0	#00000	-	15						
BSTRTN	Abs	92463	#1692F	-	219	240					
CALC10	Abs	92249	#16859	-	157	160					
CBLNK	Abs	92020	#16774	-	96	100					
CHEDIT	Ext			-	310						
CHR10	Abs	92313	#16899	-	174	178					
CHRONE	Abs	92296	#16888	-	170	57	75				
CLCATN	Abs	91898	#166FA	-	65	34	44	52	316	324	327
CLCBFR	Abs	193910	#2F576	-	12	155	247	284			
CLCBKS	Abs	92390	#168E6	-	198	37					
CLCBTS	Ext			-	137						
CLCCB	Abs	92678	#16A06	-	283	48	51	55	96	323	
CLCCD	Abs	92576	#169A0	-	251	321					
CLCCFL	Abs	91866	#166DA	-	57	35					
=CLCCFR	Abs	91927	#16717	-	70	36	88	140	144	198	268
CLCCL	Abs	91950	#1672E	-	75	38					
CLCCR	Abs	91974	#16746	-	83	39					
CLCCT	Abs	92558	#1698E	-	246	50	322				
CLCCU	Abs	92682	#16A0A	-	284	320					
CLCED	Abs	92772	#16A64	-	310	311					
CLCEDX	Abs	92618	#169CA	-	264	315	317	318	319		
CLCEOL	Abs	92202	#1682A	-	144	43	46				
=CLCET1	Abs	91762	#16672	-	30						
=CLCET2	Abs	92794	#16A7A	-	315	312					
CLCEXP	Ext			-	136						
CLCOFF	Abs	91883	#166EB	-	62	54	326				
CLCOUT	Abs	92231	#16847	-	153	53	325				
CLCSST	Abs	92024	#16778	-	98	45	47				
CLCSTK	Abs	193925	#2F585	-	12	145	213				
CLRXDS	Ext			-	68						
CNTCHR	Ext			-	220						
COMPIL	Ext			-	128						
CONFMN	Ext			-	168						
CRSTAT	Abs	91987	#16753	-	86	60	80				
DECLEN	Abs	92448	#16920	-	215						
DEFADR	Abs	194919	#2F967	-	12	277					
DSLEEP	Ext			-	63						
DSPBFS	Abs	193664	#2F480	-	12	170	228	262	278		
DSPRST	Ext			-	167	182					
DSPSTA	Abs	193653	#2F475	-	12	70	192				
DSTATE	Abs	4	#00004	-	19	141	266				
ENTRY	Abs	92335	#168AF	-	182	246	283				
ERRMSG	Ext			-	99						
EXIT10	Abs	92635	#169DB	-	271	275					
FIRSTC	Abs	193660	#2F47C	-	12	59	76	83	302		
GROWL	Abs	92522	#1696A	-	234	238					
HASH2	Ext			-	313						
HRDWAY	Abs	92465	#16931	-	220	204					
KILLKY	Ext			-	69						
LOCAL	Abs	6	#00006	-	21						
MEMBER	Ext			-	125						

MOVE1	Abs	92719	#16A2F	-	296	249	289		
MOVE2	Abs	92728	#16A38	-	299	261			
MOVEU1	Ext			-	211				
MOVEU3	Ext			-	229	306			
NEWSTM	Ext			-	244				
NOP	Abs	3	#00003	-	18				
NXTCHR	Ext			-	142				
ORIGIN	Ext			-	65	153			
PAREN	Abs	10	#0000A	-	25	267			
PARPRP	Ext			-	104				
PHNYKY	Abs	92652	#169EC	-	277	241			
PSHSTM	Ext			-	181				
REDUCE	Ext			-	98				
RENTRY	Abs	92329	#168A9	-	181	189			
RFNBFR	Abs	193915	#2F57B	-	12	183	199		
RSTATE	Abs	5	#00005	-	20	102			
RTNCC	Abs	92018	#16772	-	95	103			
S-R1-2	Abs	194699	#2F88B	-	12	111	130	215	230
SFLAGC	Ext			-	166				
SNAG	Abs	9	#00009	-	24				
SNARL	Abs	92531	#16973	-	237	233			
SST05	Abs	92030	#1677E	-	99	121			
SST10	Abs	92062	#1679E	-	107	117			
SST20	Abs	92087	#167B7	-	114	106			
SST30	Abs	92172	#1680C	-	136	126			
STAKDN	Ext			-	64				
STAKUP	Ext			-	62				
STKBAK	Ext			-	139				
STRING	Abs	1	#00001	-	16				
SYNTAX	Ext			-	101				
VAR	Abs	11	#0000B	-	26				
WIPOUT	Ext			-	309				
atnclr	Ext			-	67				
ePRMIS	Ext			-	120				
flCALC	Ext			-	165				
kcSST	Ext			-	232				
newstm	Abs	92552	#16988	-	244	66			
t-	Ext			-	105				

Input Parameters

  Source file name is AB&ED::MS

  Listing file name is AB/ED:TI:ML::-1

  Object file name is AB%ED:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                *    FFFFF  H   H  &     TTTTT  FFFFF  M   M
                *    F      H   H  & &     T    F      MM MM
                *    F      H   H  & &     T    F      M M M
                *    FFFF   HHHHH  &      T    FFFF   M M M
                *    F      H   H  & & &   T    F      M   M
                *    F      H   H  &  &    T    F      M   M
                *    F      H   H   && &   T    F      M   M
                *
                        TITLE  TRANSFORM Execution <831212.1203>
16AAE                   ABS    #16AAE
                        RDSYMB TI%EQU::MS
                        RDSYMB SB%RAM::MS

                ********************************************************************
                ********************************************************************
                **
                ** Name:    TRSFMX  -  Execute TRANSFORM Statement
                **
                ** Category:   STEXEC
                **
                ** Purpose:  Execute TRANSFORM Statement
                **
                ** Entry:    TRANSFORM Token Found
                **      P      =  0
                **      D0     @  Token immediately following TRANSFORM token
                **
                ** Exit:     To NXTSTM if no Execution Errors
                **           To BSERR if errors:
                **             Illegal Filename
                **             Illegal Transform
                **
                ** Calls:    MEMCHE,ALINFO,SKPTKN,RLINFO,DEFFIL,BSERR,NXTSTM,
                **           FSPECx,SVINFO,TRSFM-
                **
                ** Uses:
                **  Exclusive: C,D0,D1,S3(sDEST)
                **  Inclusive: All CPU registers, S11-S0, all STMT, FUNC
                **      and SCRTCH RAM, TRFMBF, SNAPBF, RSTKBF
                **
                ** Algorithm:
                **
                **  0.0 Reserve stack memory for filenames
                **      Initialize status to "Source"
                **  1.0 Try to skip INTO
                **      If successful, then
                **         If at "Source" then [source defaulted]
                **            Back up over INTO token
                **            Source file <-- current file & device
                **            Go to 2.2
                **         Else   [at file type]
                **            Read and save away dest file type
                **            If at end of line then  [destination defaulted]
                **               Dest file <-- source file & device
                **               Go to 2.2
                **      Else
```

```
                **  2.0    Parse file name (exit if error)
                **  2.2 Save file info on stack
                **      If at "Source" then Goto 1.0
                **      Call Transform utility
                **
                ********************************************************************
                ********************************************************************
                *
                *  Parse/decompile addresses
                *
 16AAE 0000 0           REL(5) =TRSFMD        Transform Decompile pointer
 16AB3 0000 0           REL(5) =TRSFMP        Transform Parse pointer

                *
                * Reserve stack memory for filenames
                * Initialize status to "Source"
                *
 16AB8 8F00 000 =TRSFMX GOSBVL =ALINFO        Allocate stack area for file info
 16ABF 445              GOC    bserr          .
 16AC2 843              ST=0   =sDEST         Status: "Source"
                *
                * Try to skip INTO
                * If successful, then
                *    If at "Source" then [source defaulted]
                *       Back up over INTO token
                *       Source file <-- current file & device
                *       Go to 2.2
                *
 16AC5 AF2   TFX100 C=0    W                  Look for INTO token
 16AC8 35FE 10E2        LC(6)  =tINTO          .
 16AD0 AF5              B=C    W               .
 16AD3 15E5             C=DAT0 6               .
 16AD7 975              ?B#C   W               File name instead?
 16ADA D2               GOYES  TFX200
 16ADC 863              ?ST=0  =sDEST          Were we looking for source file?
 16ADF B3               GOYES  TFX225          . If so, fill in current file
 16AE1 165              DO=DO+ 6               Skip INTO
                *
                *  Else   [at file type]
                *     Read and save away dest file type
                *     If at end of line then  [destination defaulted]
                *        Dest file <-- source file & device
                *
 16AE4 15E3  TFX120 C=DAT0 4                   Read dest type leaving A&D alone
 16AE8 163              DO=DO+ 4               .
 16AEB 1F1D 8F2         D1=(5) /DFTYP          . and save away
 16AF2 15D3             DAT1=C 4               .
 16AF6 817              DSRC                   Set D(S) back to device code
 16AF9 310F             LCHEX  F0              Test for EOL leaving A&D alone
 16AFD D5               B=C    A               .
```

```
106 16AFF 14E          C=DAT0 B                .
107 16B02 9ED          ?C>=B  B                EOL token?
108 16B05 D2           GOYES  TFX240            . (save source file info as dest)
109              *
110              *  Fetch file name (exit if error)
111              *  Save file info on stack
112              *  If at "Source" then
113              *     Set "Dest" status
114              *     Go to 1.0
115              *
116 16B07 8E00  TFX200 GOSUBL =fspecx          Fetch file name
          00
117 16B0D 561          GONC   TFX230           Branch if OK
118              *
119 16B10 77B1  TFX220 GOSUB  TFURLI           Release save stack area
120 16B14 8C00  bserr  GOLONG =BsErr           Error exit
          00
121              *
122 16B1A 8F00  TFX225 GOSBVL =DEFFIL          Fetch default (current) file
          000
123 16B21 501          GONC   TFX240
124              *
125 16B24 8F00  TFX230 GOSBVL =FLDEVX          Make device code explicit
          000
126 16B2B 3100         LC(2)  =eFSPEC          Load "Invalid File Spec"
127 16B2F 40E          GOC    TFX220           Branch if inexplicit dest port
128 16B32 7976  TFX240 GOSUB  svinf+           Save away file name, shifted device
129 16B36 873          ?ST=1  =sDEST           Were we at dest?
130 16B39 80           GOYES  TFX300
131 16B3B 853          ST=1   =sDEST           Set "Dest" status
132 16B3E 568          GONC   TFX100           B.E.T.
133              *
134              *  Transform file
135              *  If error, then issue error message but return to here
136              *  If "Transformed current file" then
137              *     Go to END ALL (collapse all stacks, close all files)
138              *  Else
139              *     If error, then
140              *        Go to MFERRS (Wind up at RUN loop)
141              *     Else
142              *        Go to NXTSTM (Process next statement)
143              *
144 16B41 7C30  TFX300 GOSUB  TRSFM+           Transform file
145 16B45 7285         GOSUB  CARYSV           Save carry
146 16B49 10B          R3=C
147 16B4C 590          GONC   TFX320           If no error
148 16B4F 8F00         GOSBVL =MFERR*          Issue error message, but return
          000
149 16B56 87B   TFX320 ?ST=1  =sPRGCF          Transformed current running file?
150 16B59 01           GOYES  endall
151 16B5B 11B          C=R3                    Recall error flag
152 16B5E 94A          ?C=0   S                Error?
153 16B61 F0           GOYES  mferrs           If so, continue to run loop
154 16B63 8C00         GOLONG =NXtstm          Go to next statement
          00
```

```
155 16B69 8D00  endall GOVLNG =ENDALL        Deallocate stacks
          000
156 16B70 8D00 =mferrs GOVLNG =MFERRS        Continue MFERR exit
          000
157             *
158             *
159             *^^ SR 1025-6
160             *
161             *   Bug fix for TRANSFORM copy from TAPE to RAM
162             *
163 16B77 A97   TFU040 D=C    WP
164 16B7A 20           P=     0
165 16B7C 6536         GOTO   svinfo
166 16B80 0            CON(1) =FIXSPC
167             *^^
168
```

```
169           TRSFMu  STITLE Transform Utility Routine
170           ********************************************************************
171           ********************************************************************
172           **
173           ** Name:(S) TRSFMu  -  Transform Utility Routine
174           **
175           ** Category:   EXCUTL
176           **
177           ** Purpose:
178           **      Transform a file using source/dest file info on Save
179           **      Stack.
180           **
181           ** Entry:
182           **      P      = 0
183           **      /DFTYP =  Destination file type
184           **      Save stack info set up by SVINFO as by COPYX or TRSFMX
185           **
186           ** Exit:
187           **      P       = 0
188           **      C(1-0)  =  Transformation option
189           **      C(6-2)  =  Dest file creation first parameter
190           **      C(11-7) =  Dest file creation second parameter
191           **      Save Stack info cleared from Save Stack
192           **     Carry clear:
193           **      Transform completed successfully
194           **     Carry set:
195           **      C(3-0) =  Error code.  "Syntax" if all errors were
196           **                  recoverable.
197           **
198           ** Calls:      FILFIL,EXPDEV,RDINFD/S,SVINFO,OPENF*,FINDFS,CRTF,
199           **             RAM/OM,MEMCHE,POLL,PRGFI#,LOCFI+,RPLLIN,
200           **             WRITNB, and a host of local utilities
201           **
202           ** Uses.......
203           **  Inclusive: All CPU registers, statement and function scratch,
204           **      TRFMBF, S11-S0
205           **
206           ** Stk lvls:   6   (RSTKBF: 1 plus any used by handlers)
207           **
208           ** Detail:
209           ** -------
210           **
211           **  Status Used:
212           **  ------------
213           **
214           **  Val Phase Name    Meaning
215           **  --- ----- ------  ----------------------------------------
216           **  (0) ( 1 ) sEXTDV  Source or destination is on HPIL device.
217           **
218           **  (0) ( 2 ) sTFREQ  If set, a transform is required.  Other-
219           **                    wise it is a trivial case (file is already
220           **                    desired type).
221           **
222           **  (1) ( 1 ) sUNDEF  Indicates both file names are undefined.
223           **
```

```
224             **  (1) (3,4) sTFERR  If set, a fatal error has occurred some-
225             **                    where in the transform.  User will be
226             **                    notified at end.
227             **
228             **  (2) ( 1 ) sCARD   Indicates a card device on source or dest.
229             **
230             **  (2) (3,4) sTFWNG  If set, a recoverable error has occurred
231             **                    during the transform.  This will become
232             **                    a fatal error after the transform is
233             **                    complete.
234             **
235             **  (3) (all) sDEST   If set, we are accessing the destination
236             **                    file info on the save stack.
237             **
238             **  (4) (all) sREADI  Used by RDINFO
239             **
240             **  (5) (234) sTFINP  If set, transform is in place.  Else, the
241             **                    source and dest files are distinct.
242             **
243             **  (6) ( 3 ) sTFINV  If set, we are doing an inverse transform.
244             **                    A nonrecoverable error at this point will
245             **                    cause the file to be purged.
246             **
247             **  (7) (all) sEOF    If set, an EOF has been read on input
248             **                    operation.
249             **
250             **  (8) ( 3 ) sTFEND  If set, the last line has been transformed.
251             **                    Must be used because sEOF can be set BEFORE
252             **                    the EOF is transformed.
253             **
254             **  (9) (234) sDRYRN  If set, transform results are not written
255             **                    to dest, but used to calculate required
256             **                    size of dest file.
257             **
258             **  (10)(all) sI/OBF  If set, last referenced opened file is
259             **                    external.
260             **
261             **  (11)(234) sPRGCF  If set, the transformed file was current
262             **                    file or was referenced on the stack (e.g.,
263             **                    CALLer of current file)
264             **
265             ** Algorithm:
266             **      Save return address
267             **      Initialize FIB storage and status to zero
268             **      Fill in missing file names (e.g., :TAPE INTO TEXT A)
269             **      If either filename undefined, then error exit
270             **      Expand destination device code
271             **      If dest device not specific, then error exit
272             **      Open source file (exit if error)
273             **      Save away source FIB#
274             **      Build expanded source device code
275             **      Save away source file type
276             **      Clear status
277             **      If source device = dest device
278             **      If source name = dest name
```

```
279           **        Set "Transform IN PLACE"
280           **           Store source FIB# as dest FIB#
281           **           If file is secure, then
282           **              Error exit
283           **     If dest device is HPIL then
284           **        If "Transform IN PLACE" then
285           **           Error exit
286           **     Else
287           **        If dest device is not RAM
288           **           Error exit
289           **     Find transform handler
290           **     If no handler, then
291           **     If transform required, then
292           **        Error exit
293           **     Else
294           **        If transform "In place", then
295           **           Return
296           **        Else
297           **           Copy file to destination using COPYu
298           **     If "Transform IN PLACE" then
299           **        If there is no inverse transform then
300           **           Error exit "Illegal Transform"
301           **        Release file info, clear sDRYRN
302           **        If file is current file then
303           **           Close it out and open new workfile
304           **     Else
305           **        If dest file is external then
306           **           Set "Dry Run"
307           **        Else
308           **           Search for dest file
309           **           If file found, then
310           **              Error exit "File exists"
311           **           Else
312           **              Create dest file
313           **              Open dest file
314           **              Store away dest file FIB#
315           **              Release file info
316           **     Initialize counts: NUMLINES, DESTLEN
317           **     Save away true AVMEME
318           **  3.1 Set up default output buffer
319           **     Save status
320           **     Verify minimal memory requirement
321           **     Call Transform routine
322           **     Restore status
323           **     If Error, then
324           **        If recoverable, then
325           **           Issue warning message
326           **           Set "Warning" status
327           **     If no error, then
328           **        If "Dry Run" then
329           **           Adjust destination length counter
330           **           If at EOF then
331           **              Create dest file
332           **              Open dest file
333           **              Store away dest FIB #
```

```
334          **              Clear "Dry run" status
335          **              Rewind source file
336          **            Go to 3.1  [go to next line]
337          **           Else
338          **              Read dest FIB
339          **              If transform NOT IN PLACE, then
340          **                 Set old line length to zero for insertion
341          **              Call WRITNB to copy output buffer to dest file
342          **              If fatal error, then
343          **                 Go to 3.5
344          **              Else
345          **                 Go to 3.1          [go to next line]
346          **        Else {error}
347          **           If recoverable error, then
348          **              If in "Dry run" then
349          **                 Go to code sequence to process line
350          **           Else
351          **              Set error flag
352          **              Save error code
353          **              If in "Dry run" then
354          **                 Go to 4.0
355          **              If memerr, substitute "Transform Failed" message
356          **           Issue warning message
357          **           If recovereble error, then
358          **              Set warning flag
359          **              Go to code sequence to process line
360          **           Else [It's an unrecoverable error]
361          **              If INVERSE transformation, then
362          **                 Save "Transform failed" error massage
363          **                 Go to 3.9
364          **              Else
365          **                 If transform IN PLACE, then
366          **                    Set "Inverse Transform" status
367          **                    Rewind source file
368          **                    Set up inverse transform address
369          **                    Go to 3.1
370          ** 3.7  Collapse input, output buffer
371          **        If Dry Run then
372          **           Fetch Source FIB
373          **        Else
374          **           Fetch Dest FIB
375          **           If NOT inverse transformation, then
376          **              Truncate file to current position
377          **        Rewind file, save status
378          **        Call Chain Handler on file
379          **        Call TFUSVE to hold error code, restore status
380          **        If Dry Run, then
381          **           Play it again, Sam
382          **        If error, then
383          **           Set error code
384          **  3.9    Purge destination file
385          **  4.0 Restore return address
386          **        Read source FIB#
387          **        If source FIB# not zero then
388          **           If transform IN PLACE then
```

```
389           **          Add file type and copy code to header
390           **        Close source FIB
391           **      Read dest FIB#
392           **      If dest FIB# not zero then
393           **        Close dest file
394           **      If not "Fatal error" status, then
395           **        If not "Warning" status, then
396           **          Exit successfully
397           **        Else
398           **          Set "Syntax" error code
399           **      Issue error code message
400           **      Exit with error condition
401           **
402           **
403           ** History:
404           **
405           **    Date     Programmer            Modification
406           **  --------   ----------   ---------------------------------
407           **  06/20/82      FH        Split off from TRSFMX code
408           **
409           ********************************************************************
410           ********************************************************************
411
412           *.....................................................
413           *                 Equate Local Symbols                .
414           *.....................................................
415
416           **
417           *          TRFMBF space used by TRANSFORM Utility
418           *
419           *    (The following groupings imply required adjacency)
420           *
421           *                                  -- Size --- Contents
422           /ERRCD  EQU    (TRFMBF)+0            4       Error code
423           /SFIB#  EQU    (TRFMBF)+4            2       Source FIB#
424           /DFIB#  EQU    (TRFMBF)+6            2       Dest FIB#
425           /SFTYP  EQU    (TRFMBF)+8            4       Source file type
426           /DFTYP  EQU    (TRFMBF)+12           4       Dest file type
427           /COPYC  EQU    (TRFMBF)+16           1       Dest file copy code
428
429           /STAT   EQU    (TRFMBF)+17           4       Statuses during Xform
430
431           /DLEN   EQU    (TRFMBF)+21           5       Dest file len (DESTLEN)
432           /NUMLN  EQU    (TRFMBF)+26           5       Line count (NUMLINES)
433           /LINE#  EQU    (TRFMBF)+31           5       Line #
434
435           /OPTN   EQU    (TRFMBF)+36           2       Transform Option
436           /PARM1  EQU    (TRFMBF)+38           5       File Create Parameter 1
437           /PARM2  EQU    (TRFMBF)+43           5       File Create Parameter 2
438
439           /LNLEN  EQU    (TRFMBF)+48           5       Input line length
440           /FLAG   EQU    (TRFMBF)+53           7       Free for use by handler
441
442
443           *
```

```
444            *  Equate Status Symbols (See Detail above)
445            *
446             sTFREQ EQU    0
447            *sEXTDV EQU    0
448             sTFERR EQU    1
449            *sUNDEF EQU    1
450             sTFWNG EQU    2
451            *sCARD  EQU    2
452            *sDEST  EQU    3
453            *sREADI EQU    4
454             sTFINP EQU    5
455             sTFINV EQU    6
456            *sEOF   EQU    7
457             sTFEND EQU    8
458             sDRYRN EQU    9
459            =sI/OBF EQU    10
460             sPRGCF EQU    11          NOTE: Can't be S2-S9 (used by COPYu)
461            *
462
463
464
465            ****************************************************************
466            *                                                              *
467            *                        P H A S E   I                         *
468            *                                                              *
469            *   Initialize, verify legal devices and names, open source    *
470            *                                                              *
471            *                                                              *
472            ****************************************************************
473
474            *
475            **  Save return address
476            **  Initialize FIB storage and status to zero
477            **  Fill in missing file names (e.g., :TAPE INTO TEXT A)
478            **  If either filename undefined, then error exit
479            **  Expand destination device code
480            **  If dest device not specific, then error exit
481            *
482 16B81 AF2  =TRSFM+ C=0    W              Set options to zero
483 16B84 1F9E =TRSFMu D1=(5) /OPTN          Preset file create options
          8F2
484 16B8B 15DB         DAT1=C (/PARM2)-(/OPTN)+5 .
485 16B8F AF0          A=0    W
486 16B92 1D5C         D1=(2) /ERRCD         Zero out local storage for FIB#'s,
487 16B96 159B         DAT1=A (/SFTYP)-(/ERRCD)+4 . error code, source type ONLY
488 16B9A 1DAD         D1=(2) /DLEN          Zero out dest file len, line count
489 16B9E 159E         DAT1=A (/LINE#)-(/DLEN)+5  . and line number
490 16BA2 7CE6         GOSUB  =R<rstk        Save away calling address
491 16BA6 8F00         GOSBVL =PSHUPD        Claim room for tfm hdlr address
          000
492 16BAD 8F00         GOSBVL =FILFIL        Fill in any missing file names
          000
493 16BB4 872          ?ST=1  =sCARD         CARD or PCRD?
494 16BB7 70           GOYES  TFUER2
495 16BB9 861          ?ST=0  =sUNDEF        File names defined?
```

```
496 16BBC 41          GOYES  TFU020
497
498           ************************************************************
499           *                                                          *
500           *              Error Interlude                             *
501           *                                                          *
502           *    TFUERR sets sERR and stores the error code,           *
503           *       then jumps to TFU400 to close the files.           *
504           *                                                          *
505           *    TFUER* serves the same purpose for errors that        *
506           *       occur before sTFINP and sDRYRN have been given     *
507           *       meaningful values.                                 *
508           *                                                          *
509           ************************************************************
510 16BBE 20    TFUER2 P=     0
511 16BC0 3300        LC(4)  =eFSPEC        "Illegal Filespec"
          00
512 16BC6 08    TFUER* CLRST                Clear sPRGCF, sTFINP
513 16BC8 7824  TFUERR GOSUB  TFUSEC        Store error code
514 16BCC 6B43        GOTO   TFU400
515           *
516           **     Open source file (exit if error)
517           **     Save away source FIB#
518           **     Build expanded source device code
519           **     Save away source file type
520           *
521 16BD0 77E3  TFU020 GOSUB  TFUOPS        Open source file
522 16BD4 51F         GONC   TFUER*         Error if file not found
523 16BD7 1B9C        DO=(5) /SFIB#
          8F2
524 16BDE 148         DATO=A B
525 16BE1 7836        GOSUB  LOCFIL         Locate file header
526 16BE5 D4          A=B    A              Fetch src file type from FIB
527 16BE7 131         D1=A                   .
528 16BEA 174         D1=D1+ oFTYPb          .
529 16BED 143         A=DAT1 A               .
530 16BF0 19DC        DO=(2) /SFTYP          . and save away
531 16BF4 1583        DATO=A 4               .
532 16BF8 7287        GOSUB  ?PRFIL         Check for privacy
533 16BFC 49C         GOC    TFUER*
534 16BFF 813         DSLC                  Save shifted src device in R2
535 16C02 DB          C=D    A               .
536 16C04 10A         R2=C                   .
537 16C07 7E66        GOSUB  rdinfs         Read source file info
538 16C0B 11A         C=R2                  Update explicit source device
539           *
540           *
541           *^^ SR 1026-6
542           *
543           *   Bug fix for TRANSFORM copy from TAPE to RAM
544           *   Code was:
545           *
546           *       D=C    A               .
547           *       GOSUB  svinfo
548           *
```

```
549 16C0E 23          P=     3
550 16C10 736F        GOSUB  TFU040
551           *^^
```

```
552                     EJECT
553                 ****************************************************************
554                 *                                                              *
555                 *                        P H A S E   I I                       *
556                 *                                                              *
557                 *    Determine kind of transform, find handler, set status     *
558                 *                                                              *
559                 ****************************************************************
560
561                 *
562                 *  Source and destination filenames and devices are defined!
563                 *  See if transform is in place (filenames and devices equal)
564                 *
565                 *
566                 **    Clear status
567                 **    If source device = dest device
568                 **       If source name = dest name
569                 **          Set "Transform IN PLACE"
570                 **          Store source FIB# as dest FIB#
571                 **          If file is secure, then
572                 **             Error exit
573                 *
574 16C14 08               CLRST                   Clear status
575 16C16 AF8              B=A     W               Save src file name
576 16C19 110              A=RO                     .
577 16C1C 101              R1=A                     .
578 16C1F 7D56             GOSUB   rdinfd          Read destination device
579 16C23 B06              C=C+1   P               Check for unspecified device
580 16C26 4F2              GOC     TFU100           . and deduce "In Place" if so
581 16C29 974              ?A#B    W               Dest name # src name?
582 16C2C 24               GOYES   TFU120
583 16C2E 110              A=RO
584 16C31 119              C=R1
585 16C34 EA               A=A-C   A
586 16C36 F0               ASL     A
587 16C38 8AC              ?A#0    A               Dest name # src name?
588 16C3B 33               GOYES   TFU120
589 16C3D 11A              C=R2                    Recall src device
590 16C40 EB               C=C-D   A               Compare lower 4 nibs of device code
591 16C42 F2               CSL     A                .
592 16C44 8AA              ?C=0    A               Src device = dest device?
593 16C47 F0               GOYES   TFU100           . then it's in place
594 16C49 817              DSRC                    Check for explicit port
595 16C4C 8F00             GOSBVL  =FLDEVX          .
            000
596 16C53 5A1              GONC    TFU120           . and deduce "Out of Place" if so
597 16C56 855   TFU100     ST=1    sTFINP          Set "Transform IN PLACE"
598 16C59 7DC4             GOSUB   TFURDS          Read source FIB#
599 16C5D 171              D1=D1+ (/DFIB#)-(/SFIB#)  Store as dest FIB#
600 16C60 149              DAT1=A B
601 16C63 76B5             GOSUB   LOCFIL          Locate file
602 16C67 7517             GOSUB   ?PRFI+          Check if file secure
603 16C6B 4D1              GOC     ^FUERR           . and error out if so
604                 *
605                 *  Now decide if requested transform is possible
```

```
606                *
607                **    If dest device is HPIL then
608                **       If "Transform IN PLACE" then
609                **          Error exit
610                **    Else
611                **       If dest device is not RAM
612                **          Error exit
613                *
614 16C6E 7206  TFU120 GOSUB  rdinfx          Dest device HPIL?
615 16C72 A06          C=C+C  P                .
616 16C75 471          GOC    TFU140           . branch if yes
617 16C78 A0F          D=D-1  P                Dest is :MAIN?
618 16C7B 411          GOC    TFU140           . GOYES
619 16C7E 90B          ?D=0   P                Dest is independent RAM?
620 16C81 C0           GOYES  TFU140           . error if not
621
622                ************************
623                *   Error Interlude    *
624                ************************
625 16C83 3300  TFUER3 LC(4)  =eFACCS         Error: "Illegal Access"
          00
626 16C89 6E3F  ^FUERR GOTO   TFUERR
627                ************************
628
629                *
630                **   Find transform handler
631                **   If no handler, then
632                **      If transform required, then
633                **         Error exit
634                **      Else
635                **         If transform "In place", then
636                **            Return
637                **         Else
638                **            Copy file to destination using COPYu
639                *
640 16C8D 7503  TFU140 GOSUB  TFUDTP          Read dest file type into A
641 16C91 DA           A=C    A                .
642 16C93 1C3          D1=D1- (/DFTYP)-(/SFTYP)  Read src file type into C
643 16C96 15F3         C=DAT1 4                    .
644 16C9A 7193         GOSUB  TFHDLR
645 16C9E 593          GONC   TFU200           Branch if handler found
646 16CA1 860          ?ST=0  sTFREQ           Transform NOT required?
647 16CA4 C0           GOYES  TFU180
648
649                ************************
650                *   Error Interlude    *
651                ************************
652 16CA6 3300  TFUER4 LC(4)  =eILTFM         "Illegal transform"
          00
653 16CAC 6CDF         GOTO   ^FUERR
654                ************************
655
656 16CB0 76D5  TFU180 GOSUB  Rstk<r          Restore return address
657 16CB4 8F00         GOSBVL =POPUPD         Release room for tfm hdlr address
          000
```

```
    658 16CBB 75C2          GOSUB  TFUCLS          Close source file
    659 16CBF 875           ?ST=1  sTFINP          Transform in place?
    660 16CC2 90            GOYES  TFU190
    661 16CC4 8D00          GOVLNG =COPYu          What we have here is just a copy...
              000
    662
    663                 ****************************************************************
    664                 *
    665                 *  TFURLI  -  Release Save Stack Area
    666                 *
    667                 *  Exit:
    668                 *       C(A)   =  Entry state
    669                 *       Carry clear
    670                 *
    671                 *  Uses:       D(A) plus RLINFO
    672                 *
    673 16CCB            TFU190                     What we have here is a No-op
    674 16CCB D7         TFURLI D=C     A           Save C(A)
    675 16CCD 8F00          GOSBVL =RLINFO         Release stack area
              000
    676 16CD4 DB            C=D    A               Recall C(A)
    677 16CD6 03            RTNCC
    678
    679               *
    680               **   If "Transform IN PLACE" then
    681               **      If there is no inverse transform then
    682               **         Error exit "Illegal Transform"
    683               **      Release file info, clear sDRYRN
    684               **      If file is current file then
    685               **         Close it out and open new workfile
    686               *
    687 16CD8 75F2  TFU200 GOSUB  TFURHA          Set D1 @ hdlr address storage
    688 16CDC 145           DAT1=C A               Store hdlr addr on GOSUB stack
    689 16CDF 1F5D          D1=(5) /COPYC          Store dest copy code
              8F2
    690 16CE6 1554          DAT1=C S                .
    691 16CEA 875           ?ST#0  sTFINP          Transform IN PLACE?
    692 16CED 60            GOYES  TFU210
    693 16CEF 61A0          GOTO   TFU260
    694 16CF3 7AB6  TFU210  GOSUB  TFURIA          Read inverse address
    695 16CF7 4EA           GOC    TFUER4          Error if no inverse handler
    696 16CFA 7894          GOSUB  TFUSFH          Find source file header
    697 16CFE 87A           ?ST=1  sI/OBF          External file?
    698 16D01 28            GOYES  TFUER3           . "Invalid Access" if so
    699 16D03 7395          GOSUB  D0=/B+          Save file header address in S-R1-1
    700 16D07 144           DAT0=C A                .
    701               ****************************************************
    702               *             ENTER STATUS-SAVED AREA              *
    703               ****************************************************
    704 16D0A 7672          GOSUB  TFUCLS          Close source file, save status
    705 16D0E 7A85          GOSUB  D0=/BX          D0 = File header address
    706 16D12 130           D0=A                    .
    707 16D15 7065          GOSUB  rdinfs          Find source file name
    708 16D19 AF8           B=A    W               Compare with upper case
    709 16D1C 8F00          GOSBVL =CVUCW           .
```

```
          000
  710 16D23 970          ?A=B   W              File name upper case?
  711 16D26 41           GOYES  TFU220          .
  712 16D28 7684         GOSUB  svinfo         Resave upper case name
  713 16D2C 8F00         GOSBVL =FILEF         Already a file by that name?
          000
  714 16D33 5B7          GONC   TFUER5          . (error if so)
  715 16D36 1507         DAT0=A W              Rename file to uppercase
  716 16D3A 132   TFU220 ADOEX                 Check for current file
  717 16D3D 8F00         GOSBVL =D1=CRS         .
          000
  718 16D44 8A6          ?A#C   A               .
  719 16D47 92           GOYES  TFU250          . and branch if NOT
  720 16D49 87D          ?ST=1  13             Branch if program running
  721 16D4C C0           GOYES  TFU230          .
  722 16D4E 8F00         GOSBVL =ZERPGM        Zero out program addresses if from
          000
  723 16D55 501          GONC   TFU240          . keyboard, else leave status to
  724             *                             . go to NXTSTM (BET)
  725             *
  726 16D58 7095  TFU230 GOSUB  TFUSTR         Restore status for now
  727             ******************************************************
  728 16D5C 84D          ST=0   13             Clear program running flag
  729 16D5F 85B          ST=1   sPRGCF         Set "Purged Current Running File"
  730             ******************************************************
  731 16D62 72C5         GOSUB  TFUSTS         Save status
  732 16D66 85A   TFU240 ST=1   10             Set flag for no collapse & nocat
  733 16D69 8F00         GOSBVL =EDITWF        Set up new workfile
          000
  734 16D70 7825  TFU250 GOSUB  D0=/BX         Recall file address
  735 16D74 7063         GOSUB  DE-REF         Remove file from memory references
  736 16D78 7F32         GOSUB  TFUOPS         Re-open file
  737 16D7C 583          GONC   TFUER6         Exit if error
  738 16D7F 1F9C         D1=(5) /SFIB#         Save FIB#
          8F2
  739 16D86 149          DAT1=A B               .
  740 16D89 7F55         GOSUB  TFUSTR         Restore status
  741             ******************************************************
  742             *             LEAVE STATUS-SAVED AREA                *
  743             ******************************************************
  744 16D8D 6480  ^FU480 GOTO   TFU300
  745             *
  746             **   Else
  747             **      If dest file is external then
  748             **         Set "Dry Run"
  749             **      Else
  750             **         Search for dest file
  751             **         If file found, then
  752             **            Error exit "File exists"
  753             **         Else
  754             **            Create dest file
  755             **            Open dest file
  756             **            Store away dest file FIB#
  757             **            Release file info
  758             *
```

```
759 16D91 7BE4  TFU260 GOSUB  rdinfd          Read dest file info
760 16D95 A06          C=C+C  P               Not HPIL?
761 16D98 580          GONC   TFU270           . GOYES
762 16D9B 859          ST=1   sDRYRN          Start dry run
763 16D9E 4EE          GOC    ^FU480          BET
764             ********************************************************
765             *              ENTER STATUS-SAVED AREA                 *
766             ********************************************************
767 16DA1 7385  TFU270 GOSUB  TFUSTS          Save status
768 16DA5 8F00         GOSBVL =FINDFS
          000
769 16DAC 4B1          GOC    TFU280          Branch if file not found
770
771             ************************
772             *   Error Interlude    *
773             ************************
774 16DAF 3300  TFUER5 LC(4)  =eFEXST         Error exit "File exists"
          00
775 16DB5 D7    TFUER6 D=C    A               Save error code
776 16DB7 7135         GOSUB  TFUSTR          Restore status
777 16DBB DB           C=D    A               Restore error code
778 16DBD 6A0E  ^^UERR GOTO   TFUERR
779             ************************
780
781 16DC1 849   TFU435 ST=0   sDRYRN          End dry run (entered from below)
782 16DC4 7065         GOSUB  TFUSTS          Save status
783 16DC8 74B4  TFU280 GOSUB  rdinfd          Fetch dest file info
784 16DCC 817          DSRC                    . into A, R0, and D
785 16DCF 73C1         GOSUB  TFUDTP          Fetch dest file type
786 16DD3 109          R1=C                    . into R1
787 16DD6 AF2          C=0    W
788 16DD9 1DBE         D1=(2) /PARM1          Fetch 1st file parameter (len)
789 16DDD 147          C=DAT1 A                .
790 16DE0 10A          R2=C                    . into R2
791 16DE3 174          D1=D1+ (/PARM2)-(/PARM1) Fetch 2nd file parameter
792 16DE6 147          C=DAT1 A                . into R3
793 16DE9 10B          R3=C                    .
794 16DEC 8E00         GOSUBL =CRTF           Create file
          00
795 16DF2 42C          GOC    TFUER6
796 16DF5 853          ST=1   =sDEST          Open dest file
797 16DF8 72C1         GOSUB  TFUOPN
798 16DFC D8           B=A    A               Save FIB#
799 16DFE 7202         GOSUB  TFUSVE          Restore status
800             ********************************************************
801             *              LEAVE STATUS-SAVED AREA                 *
802             ********************************************************
803 16E02 460          GOC    TFU290          If no error
804 16E05 6A01         GOTO   TFU385          Purge dest file, finish up
805 16E09 1DBC  TFU290 D1=(2) /DFIB#          Store away dest FIB#
806 16E0D D4           A=B    A               .
807 16E0F 149          DAT1=A B               .
```

```
808                     EJECT
809                 ****************************************************************
810                 *                                                              *
811                 *                      P H A S E   I I I                       *
812                 *                                                              *
813                 *    Transform file line by line, calling handler each time    *
814                 *                                                              *
815                 ****************************************************************
816
817                 *
818                 **   Initialize counts: NUMLINES, DESTLEN
819                 **   Save away true AVMEME
820                 *
821 16E12 AF0    TFU300 A=0    W              Initialize counters and Line#:
822 16E15 1FAD          D1=(5) /DLEN           . DESTLEN = NUMLINES = LINE# = 0
          8F2
823 16E1C 159E          DAT1=A (/LINE#)-(/DLEN)+5   .
824 16E20 848           ST=0   sTFEND          Set "In progress" status *< REMOVE
825                 *
826                 **  3.1 Set up default output buffer
827                 **      Save status
828                 **      Verify miminal memory requirement
829                 **      Call Transform routine
830                 **      Restore status
831                 *
832 16E23 1BFD   TFU310 D0=(5) /NUMLN          Increment number of lines
          8F2
833 16E2A 142           A=DAT0 A                .
834 16E2D E4            A=A+1  A                .
835 16E2F 866           ?ST=0  sTFINV           . NOT in inverse transformation?
836 16E32 B0            GOYES  TFU315           .
837 16E34 CC            A=A-1  A                . Decrement
838 16E36 CC            A=A-1  A                .  .
839 16E38 8A8           ?A=0   A                . No more lines to de-transform?
840 16E3B D3            GOYES  tfu370           .
841 16E3D 140    TFU315 DAT0=A A                . Store new line count
842 16E40 878           ?ST=1  sTFEND           Last line transformed?
843 16E43 53            GOYES  tfu370
844 16E45 7B42          GOSUB  TFLINE           Transform line
845                 ****************************************************
846                 *             ENTER STATUS-SAVED AREA              *
847                 ****************************************************
848 16E49 77B1          GOSUB  TFUSVE           Save error, restore status
849                 ****************************************************
850                 *             LEAVE STATUS-SAVED AREA              *
851                 ****************************************************
852
853                 *
854                 **      If Error, then
855                 **         If recoverable, then
856                 **            Issue warning message
857                 **            Set "Warning" status
858                 **      If no error, then
859                 **         If "Dry Run" then
860                 **            Adjust destination length counter
```

```
                **          If at EOF then
                **             Create dest file
                **             Open dest file
                **             Store away dest FIB #
                **             Clear "Dry run" status
                **             Rewind source file
                **          Go to 3.1  [go to next line]
                *
16E4D 5D0           GONC   TFU320          Branch if NO error
16E50 852           ST=1   sTFWNG          Set warning flag
16E53 94A           ?C=0   S               Unrecoverable error?
16E56 84            GOYES  TFU350
16E58 5A5           GONC   TFU360          BET - issue warning message
16E5B 869   TFU320  ?ST=0  sDRYRN          DRY RUN NOT in progress?
16E5E E1            GOYES  TFU330
16E60 7204          GOSUB  oblcmp          Compute output line length
16E64 1FAD          D1=(5) /DLEN           Adjust  DESTLEN
      8F2
16E6B 147           C=DAT1 A                .
16E6E C2            C=A+C  A                .
16E70 145           DAT1=C A                .
16E73 868           ?ST=0  sTFEND          NOT EOF?
16E76 DA            GOYES  TFU310           .
16E78 6260  tfu370  GOTO   TFU370          Prepare to play it again, Sam
                *
                **      Else
                **         Read dest FIB
                **         If transform NOT IN PLACE, then
                **            Set old line length to zero for insertion
                **         Call WRITNB to copy output buffer to dest file
                **         If error, then
                **            Go to 3.5
                **         Else
                **            Go to 3.1
                *
16E7C 7561  TFU330  GOSUB  TFURLL          Read line length into R3
16E80 7292          GOSUB  TFURD+          Read dest FIB#, set C(A)=0
16E84 875           ?ST=1  sTFINP          Transform IN PLACE?
16E87 50            GOYES  TFU340
16E89 10B           R3=C                   Set up for insertion
16E8C 7894  TFU340  GOSUB  TFUSTS          Save status
                ************************************************
                *            ENTER STATUS SAVED AREA             *
                ************************************************
16E90 7796          GOSUB  WRITNB          Write line to file
16E94 847           ST=0   sEOF            Clear possible EOF after write
16E97 7961          GOSUB  TFUSVE          Save error, restore status
                ************************************************
                *            LEAVE STATUS SAVED AREA             *
                ************************************************
16E9B 578           GONC   TFU310          If not fatal error, continue
                **
                **  If error, then
                **     If recoverable error, then
                **        If in "Dry run" then
```

```
915                **          Go to code sequence to process line
916                **       Else
917                **          Set error flag
918                **          Save error code
919                **          If in "Dry run" then
920                **             Go to 4.0
921                **          If memerr, substitute "Transform Failed" message
922                **       Issue warning message
923                **       If recovereble error, then
924                **          Set warning flag
925                **          Go to code sequence to process line
926                **
927 16E9E 842    TFU350 ST=0   sTFWNG         Clear warning flag
928 16EA1 7F41          GOSUB  TFUSEC         Save error code, set sTFERR
929 16EA5 879           ?ST=1  sDRYRN         If "Dry run"
930 16EA8 07            GOYES  TFU400
931 16EAA 3300          LC(4)  =eTFFLD        Display "Transform Failed"
          00
932 16EB0 108           RO=C                   .
933 16EB3 71F3   TFU360 GOSUB  TFUWRN         Issue warning message
934 16EB7 872           ?ST=1  sTFWNG         Recoverable error?
935 16EBA 1A            GOYES  TFU320
936                **
937                **
938                **       THIS IS THE "BAD NEWS" SECTION OF CODE
939                **
940                **
941                **       Else [It's an unrecoverable error]
942                **          If INVERSE transformation, then
943                **             Save "Transform failed" error massage
944                **             Go to 3.9
945                **          Else
946                **             If transform IN PLACE, then
947                **                Set "Inverse Transform" status
948                **                Rewind source file
949                **                Set up inverse transform address
950                **                Go to 3.1
951                **
952 16EBC 876           ?ST=1  sTFINV         Inverse transform?
953 16EBF 55            GOYES  TFU390
954 16EC1 865           ?ST=0  sTFINP         Transform NOT IN PLACE?
955 16EC4 05            GOYES  TFU390
956 16EC6 856           ST=1   sTFINV         Set "Inverse Transform" status
957 16EC9 848           ST=0   sTFEND         Set "In Progress" status
958 16ECC 79C4          GOSUB  TFURWS         Rewind source file, read inv tfm
959 16ED0 7DF0          GOSUB  TFURHA         Store as transform hdlr address
960 16ED4 145           DAT1=C A               .
961 16ED7 6B4F          GOTO   TFU310
962                *
963                **       Else
964                **  3.7     Collapse input, output buffer
965                **          If Dry Run then
966                **             Fetch Source FIB
967                **          Else
968                **             Fetch Dest FIB
```

```
                **              If NOT inverse transformation, then
                **                 Truncate file to current position
                **           Rewind file, save status
                **           Call Chain Handler on file
                **           Call TFUSVE to hold error code, restore status
                **           If Dry Run, then
                **              Play it again, Sam
                **           If error, then
                **              Set error code
                **  3.9         Purge destination file
                *
16EDB 7442  TFU370 GOSUB  TFURDC        Read src FIB, collapse buffers
16EDF 879          ?ST=1  sDRYRN        Dry Run?
16EE2 F0           GOYES  TFU380
16EE4 7032         GOSUB  TFURDD        Fetch dest FIB into A and R4
16EE8 876          ?ST=1  sTFINV        Inverse transform just completed?
16EEB 60           GOYES  TFU380
16EED 7FD4         GOSUB  TRCFI#        Truncate file
16EF1 78A4  TFU380 GOSUB  TFURWD        Rewind file, read inv tfm addr
16EF5 1C9          D1=D1- 10            Read chain handler address
16EF8 7FB4         GOSUB  RRADD          .
16EFC 77C1         GOSUB  GOSUBI        Call finish-up routine
16F00 7001         GOSUB  TFUSVE        Restore status preserving error
16F04 4B0          GOC    TFU385        If error
16F07 869          ?ST#1  sDRYRN        NOT Dry Run?
16F0A E0           GOYES  TFU400
16F0C 64BE         GOTO   TFU435        Play it again, Sam
16F10 70E0  TFU385 GOSUB  TFUSEC        Set fatal error code
                *
                *  I mean, this Transform REALLY failed...
                *
16F14 7B24  TFU390 GOSUB  TFUPGD        Purge source/dest file
```

```
1001                    EJECT
1002                ****************************************************************
1003                *                                                              *
1004                *                        P H A S E   I V                       *
1005                *                                                              *
1006                *    Close files, release resources, set up exit condiitons    *
1007                *                                                              *
1008                ****************************************************************
1009
1010                *
1011                **  4.0 Restore return address
1012                **      Read source FIB#
1013                **      If source FIB# not zero then
1014                **         If transform IN PLACE then
1015                **           Add file type and copy code to header
1016                **         Close source FIB
1017                **      Read dest FIB#
1018                **      If dest FIB# not zero then
1019                **         Close dest file
1020                *
1021 16F18 7E63  TFU400 GOSUB  Rstk<r          Restore return address
1022 16F1C 8F00         GOSBVL =POPUPD        Release hdlr address storage
            000
1023 16F23 74AD         GOSUB  TFURLI         Release save stack info
1024 16F27 7FF1         GOSUB  TFURDS         Read source FIB#
1025 16F2B 968          ?A=0   B              No FIB#?
1026 16F2E 03           GOYES  TFU440
1027 16F30 865          ?ST=0  sTFINP         Transform NOT in place?
1028 16F33 72           GOYES  TFU420
1029 16F35 871          ?ST=1  sTFERR         Fatal error condition?
1030 16F38 62           GOYES  TFU440
1031 16F3A 7C52         GOSUB  LOCFHD         Set D0 to file type in file header
1032 16F3E 16F          D0=D0+ =oFTYPh         .
1033 16F41 7150         GOSUB  TFUDTP         Set C to dest data type
1034 16F45 15C3         DAT0=C 4              Update file type
1035 16F49 173          D1=D1+ (/COPYC)-(/DFTYP)  Update file copycode
1036 16F4C 14B          A=DAT1 B                      .
1037 16F4F 164          D0=D0+ (oFLAGh)-(oFTYPh)+1    .
1038 16F52 1580         DAT0=A 1                      .
1039 16F56 6700         GOTO   TFU440
1040 16F5A 7A20  TFU420 GOSUB  TFUCLF         Close file, preserve status
1041 16F5E 74B1  TFU440 GOSUB  TFURD+         Read dest FIB#, set C(A) = 0
1042 16F62 14D          DAT1=C B              Zero out FIB#
1043 16F65 96C          ?A#0   B              Dest FIB# present?
1044 16F68 2F           GOYES  TFU420          . Go close it
1045                *
1046                **      If not "Fatal error" status, then
1047                **         If not "Warning" status, then
1048                **            Exit successfully
1049                **         Else
1050                **            Set "Syntax" error code
1051                **      Issue error code message
1052                **      Exit with error condition
1053                *
1054 16F6A 1D5C         D1=(2) /ERRCD         Read error code
```

```
1055 16F6E 15F3        C=DAT1 4                .
1056 16F72 871         ?ST=1  sTFERR           Return fatal error?
1057 16F75 00          RTNYES
1058 16F77 3300        LC(4)  =eSYNTX          Load default error code
           00
1059 16F7D 872         ?ST=1  sTFWNG           Warning status?
1060 16F80 00          RTNYES
1061 16F82 03          RTNCC                    .
```

```
1062                    STITLE Miscellaneous Support Routines
1063                ****************************************************************
1064                *
1065                *  TFUCLS, TFUCLF  -  Close (Source) File Preserving Status
1066                *
1067                *  Entry:
1068                *       A(B)   =  FIB# (TFUCLF only; Source FIB used for TFUCLS)
1069                *
1070                *  Exit:       Status preserved
1071                *
1072 16F84 72A1    TFUCLS GOSUB  TFURDS          Read source FIB
1073 16F88 7C93    TFUCLF GOSUB  TFUSTS          Save status
1074 16F8C 8E00            GOSUBL =CLOSEF        Close file
           00
1075 16F92 6953            GOTO   TFUSTR         Restore status
1076
1077                ****************************************************************
1078                *
1079                *  TFUDTP, TFUDT-  -  Fetch (Dest) File Type
1080                *
1081                *  Entry:
1082                *     TFUDTP:   None
1083                *     TFUDT-:   D1  @  File type save area
1084                *
1085                *  Exit:
1086                *     D1    @  File type save area
1087                *     C(A)  =  File type, zero filled
1088                *     Carry preserved
1089                *
1090                *  Uses:  D1, C(A)
1091                *
1092 16F96 1F1D    TFUDTP D1=(5) /DFTYP
           8F2
1093 16F9D D2      TFUDT- C=0    A
1094 16F9F 15F3            C=DAT1 4
1095 16FA3 01              RTN
1096                ****************************************************************
1097                *
1098                * Name:       TFUEOF, TFUCCK  -  Check for EOF or Error
1099                *
1100                * Category:   LOCAL
1101                *
1102                * Purpose:
1103                *       TFUEOF locates the file and falls into TFUCCK.
1104                *       TFUCCK returns to its callers caller if error (carry
1105                *       set) or if sEOF set (carry clear, C(A) is returned = 0).
1106                *
1107                * Entry:
1108                *      TFUEOF:
1109                *       R4(14-13) =  FIB#
1110                *      TFUCCK:
1111                *       C(A)   =  Error code if error set.
1112                *       Carry  =  Set if error.
1113                *       sEOF   =  Set if last operation was not performed due to
1114                *                 EOF.  This is a successful return condition.
```

```
1115            *
1116            * Exit: TO CALLER'S CALLER if sEOF or Carry set on entry:
1117            *       Carry set: Error
1118            *        C(A)   =  Error code
1119            *       Carry clear: sEOF set on entry
1120            *        C(A)   =  0
1121            *      OTHERWISE RETURN TO CALLER:
1122            *        C(A)   =  Entry state
1123            *        sEOF   =  Clear
1124            *        Carry  =  Clear
1125            *
1126            * Calls:        None.
1127            *
1128            * Uses.......
1129            *  Inclusive: B(A),C(A)(TFUCCK); LOCFI# usage (TFUEOF)
1130            *
1131            * Stk lvls:    0
1132            *
1133            ****************************************************************
1134 16FA5 7072  TFUEOF GOSUB  LOCFI#
1135 16FA9 490   TFUCCK GOC    TFUE20          If error
1136 16FAC 867          ?ST=0  sEOF            Return if no EOF
1137 16FAF 34           GOYES  TFURcc
1138 16FB1 D2           C=0    A               Zero C(A) for null line
1139 16FB3 D5   TFUE20 B=C    A
1140 16FB5 07           C=RSTK                 Return to caller's caller
1141 16FB7 D9           C=B    A                .
1142 16FB9 01           RTN
1143
1144            ****************************************************************
1145            *
1146            * Name:        TFUOPN, TFUOPS  -  Open File From Saved Info
1147            *
1148            * Category:    LOCAL
1149            *
1150            * Purpose:
1151            *       Open source file from SAVSTK info.  TFUOPS opens the
1152            *       source file; TFUOPN opens source or destination.
1153            *
1154            * Entry:
1155            *       P      =  0
1156            *       SAVSTK info contains source/dest file names.
1157            *       sDEST  =  Set iff source file is to be open (TFUOPN only)
1158            *
1159            * Exit:
1160            *       P      =  0
1161            *      Carry set: Successful open
1162            *       A(B)   =  FIB#
1163            *      Carry clear: Open failed
1164            *       C(3-0) =  Error code
1165            *
1166            * Calls:       OPENF*, RDINFO
1167            *
1168            * Uses.......
1169            *  Inclusive: All CPU registers, Statement, Function scratch
```

```
1170            *
1171            * Stk lvls:   7
1172
1173            *
1174            ****************************************************************
1175
1176 16FBB 843   TFUOPS ST=0   =sDEST
1177 16FBE 71C2  TFUOPN GOSUB  rdinfo         Read file info
1178 16FC2 817          DSRC                  Set up device info
1179 16FC5 AF2          C=0    W              Zero out external file entry conds
1180 16FC8 10A          R2=C
1181 16FCB 8C00         GOLONG =OPENF*
          00
1182
1183            ****************************************************************
1184            *
1185            *  TFURHA  -  Read Transform Handler Address
1186            *
1187            *  Entry:
1188            *       None.
1189            *
1190            *  Exit:
1191            *       D1     @  Top address on GOSUB stack, which is assumed
1192            *                    to be transform handler address.
1193            *       A(A)   =  Transform handler address
1194            *
1195            *  Uses:       A(A),D1
1196            *
1197 16FD1 1F3A  TFURHA D1=(5) =GSBSTK        Fetch start of GOSUB stack
          5F2
1198 16FD8 143          A=DAT1 A               .
1199 16FDB E4           A=A+1  A               Space over type nib
1200 16FDD 131          D1=A                   Read address
1201 16FE0 143          A=DAT1 A               .
1202 16FE3 01           RTN
1203
1204            ****************************************************************
1205            *
1206            *  TFURLL  -  Read Line Length into R3
1207            *
1208            *  Entry:
1209            *       /LNLEN  =  Line length
1210            *
1211            *  Exit:
1212            *       C(A)    =  Line length
1213            *       R3      =  Line length
1214            *       Carry clear
1215            *
1216            *  Uses:  D1, C(A), R3
1217            *
1218            *
1219 16FE5 1F5F  TFURLL D1=(5) /LNLEN
          8F2
1220 16FEC 147          C=DAT1 A
1221 16FEF 10B          R3=C
```

```
1222 16FF2 03    TFURcc RTNCC
1223
1224             ****************************************************************
1225             *
1226             *  TFUSEC  -  Store error code in /ERRCD, set error flag
1227             *
1228             *  Entry:
1229             *       C(3-0) =  Error code
1230             *
1231             *  Exit:
1232             *       C(3-0) =  Error code
1233             *       Carry  =  Entry state
1234             *       sTFERR =  Set
1235             *
1236 16FF4 1F5C  TFUSEC D1=(5) /ERRCD          Store error code
          8F2
1237 16FFB 15D3         DAT1=C 4               .
1238 16FFF 851           ST=1   sTFERR          Set error flag
1239 17002 01            RTN                    Return with carry preserved
1240
1241
1242             ****************************************************************
1243             *
1244             *  TFUSVE  -  Save error setting in R0, restore status
1245             *
1246             *  Entry:
1247             *       P      =  0
1248             *       Carry: Set if error
1249             *       C(3-0) =  Error code if error
1250             *       sEOF   =  Set if eof
1251             *       /STAT  =  Prior status
1252             *
1253             *  Exit:
1254             *       P      =  0
1255             *       Carry  =  Entry state
1256             *       C(S)   =  F if carry set on entry, else 0
1257             *       C(A)   =  Error code
1258             *       R0(S)  =  0 if carry set on entry, else 1
1259             *       R0(A)  =  Error code
1260             *       D1     =  /STAT
1261             *       Status =  Prior status
1262             *       sTFEND =  Set if sEOF set on entry, else prior setting
1263             *
1264 17004 F2    TFUSVE CSL    A                Isolate error code in C(A)
1265 17006 F6           CSR    A                .
1266 17008 108          R0=C                    Save error codes if any
1267 1700B 7CB0         GOSUB  CARYSV          Save carry
1268 1700F 3208         LC(3)  2^(sEOF)        Merge sEOF status with old status
          0
1269 17014 1F6D         D1=(5) /STAT            .
          8F2
1270 1701B 7DE2         GOSUB  =STATR+          .
1271 1701F 867          ?ST=0  =sEOF           Was the EOF NOT transformed?
1272 17022 50           GOYES  TFUSV1
1273 17024 858          ST=1   sTFEND          Set "End" status
```

```
1274 17027 A4E   TFUSV1 C=C-1  S            Reset carry
1275 1702A 118          C=R0                Recall error return
1276 1702D 01           RTN
1277
```

```
1278            TFHDLR  STITLE Find Transform Handler
1279            ********************************************************************
1280            ********************************************************************
1281            **
1282            ** Name:(S) TFHDLR  -  Find Transform Handler
1283            **
1284            ** Category:   FILUTL
1285            **
1286            ** Purpose:
1287            **      Find the address of a transform handler capable of
1288            **      reading and transforming lines of the source type into
1289            **      lines of the destination file type.
1290            **
1291            ** Entry:
1292            **      P       =  0
1293            **      A(A)    =  Destination file type
1294            **      C(A)    =  Source file type
1295            **      S5      =  Set if transformation is IN PLACE (sTFINP)
1296            **
1297            ** Exit:
1298            **      P       =  0
1299            **      S5      =  Preserved (sTFINP)
1300            **     Carry clear: [Transform handler found]
1301            **      S0      =  Set if transform requires a handler (sTFREQ)
1302            **      C(A)    =  Destination file copy code
1303            **      C(S)    = Transform handler address
1304            **     Carry set:
1305            **      Indicates that a transform handler NOT found, or that
1306            **        the source and destination file types are the same
1307            **        and no LEX file declared that a handler was needed
1308            **        (in this case, S0 will be clear; transform can be
1309            **        handled by COPY or by doing nothing if IN PLACE).
1310            **
1311            ** Calls:      FPOLL
1312            **
1313            ** Uses.......
1314            **  Inclusive: A,B,C,R0,D0,D1
1315            **
1316            ** Stk lvls:   5
1317            **
1318            ** History:
1319            **
1320            **    Date     Programmer              Modification
1321            **  --------   ----------   ---------------------------------
1322            **  04/01/83      FH        Derived from in-line code
1323            **
1324            ********************************************************************
1325            ********************************************************************
1326            *
1327            ********************************************************************
1328            ********************************************************************
1329            **
1330            ** Name:(S) pTRFMx  -  Poll for TRANSFORM Execution
1331            **
1332            ** Category:   POLL
```

```
1333            **
1334            ** Type:       FPOLL
1335            **
1336            **
1337            ** Purpose:
1338            **      Ask for an address to call for line-by-line transforma-
1339            **      tion, and a similar address to call for line-by-line
1340            **      inverse transformation should that become necessary.
1341            **      The interface for these routines is defined in the
1342            **      Detail below.
1343            **
1344            ** Should poll be "Handled" (return with XM=0)?:
1345            **      Yes.
1346            **
1347            ** Meaning of "Handling" Poll (what does code do if handled?):
1348            **      The required information is present in the registers.
1349            **
1350            ** Entry conditions for handler (registers, ST, RAM, etc.):
1351            **      R0(A)  =  Source file type
1352            **      R1(A)  =  Destination file type
1353            **      S0     =  Set if dest type # source type, means
1354            **                that a transform IS required (sTFREQ)
1355            **      S5     =  Set if transform is in place (sTFINP)
1356            **      /OPTN  =  TRANSFORM option set by extended TRANSFORM
1357            **                parse (or zero if mainframe parse), as in:
1358            **                    TRANSFORM F INTO DATA FF,R
1359            **                where R means random I/O records (no overlap)
1360            **                See detail below for address of /OPTN
1361            **      /PARM1,
1362            **      /PARM2 =  TRANSFORM destination file create parameters
1363            **                set by extended TRANSFORM parse (or zero if
1364            **                mainframe parse), as in:
1365            **                    TRANSFORM F INTO DATA FF,R,50,128
1366            **                where /PARM1 =  50 = number of records,
1367            **                      /PARM2 = 128 = record size
1368            **                See detail below for addresses of /PARM1 and
1369            **                /PARM2
1370            **      P      =  0
1371            **      Carry  =  Set on entry
1372            **      B[A]   =  Poll number
1373            **      HEX mode
1374            **
1375            ** Normal exit conditions from handler if handled (ST, RAM,
1376            ** registers, etc.):
1377            **      R0(A)  = Address of handler routine which can read
1378            **                one line in from source and transform it.
1379            **                See Detail below for handler interface.
1380            **               ** 5 nibbles before this address is stored the
1381            **                relative address of handler routine which can
1382            **                read one line in from source and transform it
1383            **                in the INVERSE direction; 0 if none exists.
1384            **                Interface is same as that of a normal handler
1385            **                routine.  See Detail for handler interface.
1386            **               ** 10 nibbles before this address is stored
1387            **                the relative address of a routine which will
```

```
1388            **                finish the fully transformed destination file
1389            **                before it is closed (e.g., to chain a BASIC
1390            **                file in RAM before leaving it); 0 if no such
1391            **                routine is needed.  See Detail below for the
1392            **                finish-up routine interface.
1393            **       R0(S)  =  Copy code of destination file type
1394            **       S5     =  Entry condition (sTFINP)
1395            **       S0     =  1 if transform handler routines must be called
1396            **                   to perform transformation (even though source
1397            **                   and dest file types may be the same)
1398            **              =  0 if no transform handler routines need to be
1399            **                   called (source file and dest file type must
1400            **                   be the same)
1401            **       HEX mode
1402            **       XM     =  0
1403            **
1404            ** Normal exit conditions from handler if not handled (ST, RAM,
1405            ** registers, etc.):
1406            **       Entry conditions preserved
1407            **       HEX mode
1408            **       XM     =  1
1409            **
1410            ** Available subroutine levels:
1411            **       3
1412            **
1413            ** What registers/RAM may be used if handled?:
1414            **       A-D, D0, D1, P, R0, R1, R2, S0
1415            **
1416            ** What registers/RAM may be used if not handled?:
1417            **       A-C, D[15-5] D0, D1, P, R2
1418            **
1419            **
1420            ** Special memory/pointer considerations (are pointers funny?):
1421            **       No.
1422            **
1423            ** Envisioned application(s):
1424            **       Conversion between OEM file types and TEXT(LIF1) for
1425            **       purposes of listing and interchange.
1426            **
1427            ** History:
1428            **
1429            **    Date      Programmer              Modification
1430            **  --------   ----------   ---------------------------------
1431            **  05/27/83       FH       Added new documentation header.
1432            **
1433            ********************************************************************
1434            **
1435            ** DETAIL:
1436            **
1437            **           INTERFACE TO TRANSFORM HANDLER ROUTINE
1438            **           -------------------------------------
1439            **
1440            ** Purpose:
1441            **       Read line from source file, transform it into destination
1442            **       type and leave it in output buffer.  No messages should
```

```
1443            **      be directly issued by this routine.
1444            **
1445            ** Entry:
1446            **      R4(15,14)     =  Source FIB#
1447            **      Input, output buffers collapsed to SYSEN
1448            **      At least 150 bytes + LEEWAY available memory guaranteed
1449            **      /LINE#        =  0 or previously returned BCD line #
1450            **      /SFIB#        =  Source FIB#
1451            **      /OPTN         =  Option from extended TRANSFORM statement
1452            **                       execution; 0 if from normal TRANSFORM
1453            **      /PARM1,/PARM2 = Destination file create parameters from
1454            **                       extended TRANSFORM statement execution;
1455            **                       0 if from normal TRANSFORM
1456            **      P             =  0
1457            **
1458            ** Exit:
1459            **      OUTBS  @  Start of transformed line.  If original line
1460            **                was copied into available memory start, OUTBS
1461            **                may point immediately after the original line.
1462            **                Must be collapsed to /SYSEN if fatal error.
1463            **      AVMEMS @  End of transformed line unless fatal error.
1464            **                Must be collapsed to /SYSEN if fatal error.
1465            **      S7     =  1 iff end of file found on source file (sEOF)
1466            **      /LNLEN =  Full length in nibs of input line.  Unneeded
1467            **                if fatal error.
1468            **      /LINE# =  BCD line number of current line.  Used in
1469            **                reporting error messages.  If sequential
1470            **                line number is to be used, set to 0.
1471            **      P      =  0
1472            **     Carry clear:  Successful transformation
1473            **     Carry set:    Error occurred
1474            **      C(3-0) =  Error code
1475            **      C(S)   =  0 if error was fatal (unrecoverable).
1476            **             #  0 if error was recoverable.
1477            **
1478            ** Allowed to use.......
1479            **      All CPU registers, S0-S11, S13, Statement and Function
1480            **      scratch RAM, SNAPBF, RSTKBF, /LNLEN, /LINE#, /FLAG,
1481            **      INBS, OUTBS, AVMEMS
1482            **
1483            ** Stk lvls:   6 (max)
1484            **
1485            ****************************************************************
1486            **
1487            **      INTERFACE TO TRANSFORM HANDLER FINISH-UP ROUTINE
1488            **
1489            ** Purpose:
1490            **      To finish up the destination BASIC file after all TEXT
1491            **      lines have been transformed into BASIC.  There are
1492            **      several cases to be dealt with:
1493            **
1494            **     End of a Dry Run (always out-of-place transform):
1495            **      If the destination file is on an external medium, a
1496            **      first pass or "dry run" is conducted without creating
1497            **      the dest file, in order to determine its necessary data
```

```
1498          **      size.  This routine calculates the needed parameters
1499          **      to create the file (see CRTF utility), and stores them
1500          **      in /PARM1 and /PARM2.
1501          **
1502          **     End of a Normal Run, Out-of-Place Transform:
1503          **      If the destination file type requires a subheader or
1504          **      Implementation field, it must be properly initialized
1505          **      since CRTF stored a default value in it when the file
1506          **      was created (see CRTF utility).  For example, if the
1507          **      destination is a BASIC file, hex value 0000000000F
1508          **      must be written to the header to indicate that the link
1509          **      chain heads have not been computed for this file (file
1510          **      has not been "chained").  If the file is in memory, the
1511          **      the proper link chain heads are computed and written to
1512          **      the subheader.  File data, such as links between sub-
1513          **      programs in BASIC files, may need to be updated.
1514          **
1515          **     End of a Normal Run, In-Place Transform:
1516          **      If the source file type had a subheader or Implementa-
1517          **      tion field, it must be removed.  If the destination
1518          **      file type requires a subheader or Implementation field,
1519          **      it must be inserted after the file header and set to
1520          **      the proper value.  File data, such as links between
1521          **      subprograms in BASIC files, may need to be updated.
1522          **
1523          **     End of an Inverse Transformation (Always In-Place):
1524          **      If the source file type had a subheader or Implementa-
1525          **      tion field, it it is still there but may need to be
1526          **      updated to reflect the new state.  File data, such as
1527          **      links between subprograms in BASIC files, may need
1528          **      to be updated.
1529          **
1530          ** Entry:
1531          **      Output buffer collapsed (OUTBS, AVMEMS point to SYSEN)
1532          **      R4(15-14) =  FIB# of destination file. Each line of the
1533          **                   source file (including EOF) has been read,
1534          **                   transformed, and written to the dest file.
1535          **                   The End of Data field in the FIB is set to
1536          **                   this new end of file and, if the dest file
1537          **                   is in memory, any excess nibs beyond the
1538          **                   end of file have been removed from the
1539          **                   file chain.  File is now rewound.
1540          **      S5      =  1 iff transformation is in place (sTFINP)
1541          **      S6      =  1 iff at end of inverse transformation
1542          **                   (sTFINV)
1543          **      S9      =  1 iff at end of dry run (sDRYRN)
1544          **      P       =  0
1545          **
1546          ** Exit:
1547          **      P       =  0
1548          **     Carry clear:
1549          **      No error
1550          **     Carry set:
1551          **      C(3-0) =  Error code (will be treated as fatal error,
1552          **                with no possibility of recovering dest file)
```

```
1553                ;*
1554                ;* Uses.......
1555                ;*  Inclusive: May use any CPU register, S10
1556                ;*
1557                ;* Stk lvls:   6 (max)
1558                ;*
1559                ;*****************************************************************
1560                ;*
1561                ;*       TRFMBF FIELDS USED BY TRANSFORM ROUTINES
1562                ;*       ---------------------------------------
1563                ;*
1564                ;*           TRFMBF        Set by
1565                ;*  Symbol   Offset  Size   User*  Contents
1566                ;*  ------   ------  ----   ----   ----------------------
1567                ;*  /ERRCD      0      4           Error code
1568                ;*  /SFIB#      4      2           Source FIB#
1569                ;*  /DFIB#      6      2           Dest FIB#
1570                ;*  /SFTYP      8      4           Source file type
1571                ;*  /DFTYP     12      4           Dest file type
1572                ;*  /COPYC     16      1           Dest file copy code
1573                ;*  /STAT      17      4           Statuses during Xform
1574                ;*  /DLEN      21      5           Dest file len (DESTLEN)
1575                ;*  /NUMLN     26      5           Line count (NUMLINES)
1576                ;*
1577                ;*  /LINE#     31      5      H    Line #
1578                ;*  /OPTN      36      2      X    Transform Option
1579                ;*  /PARM1     38      5      X    File Create Parameter 1
1580                ;*  /PARM2     43      5      X    File Create Parameter 2
1581                ;*  /LNLEN     48      5      H    Input line length
1582                ;*  /FLAG      53      7      H    Free for use by handler
1583                ;*
1584                ;*  * Where 'H' indicates the field is set by the handler,
1585                ;*    and   'X' indicates the field is set by the extended
1586                ;*    TRANSFORM execution routine
1587                ;*
1588                ;*****************************************************************
1589                ;*****************************************************************
1590 1702F 850   =TFHDLR ST=1   sTFREQ          Presume transform required
1591 17032 8A6          ?C#A   A               Source type # Dest type?
1592 17035 50           GOYES  TFHD20
1593 17037 840          ST=0   sTFREQ          Status: "Transform NOT required"
1594 1703A 108   TFHD20 R0=C                   Save source , dest types R0,R1
1595 1703D 101          R1=A                    .
1596                *
1597                *  Poll for Transform handler address
1598                *  If NOT handled then
1599                *     Test for mainframe types
1600                *     If no match, then error exit
1601                *
1602 17040 8E00         GOSUBL =FPOLL          Fast poll for TRANSFORM handler
           00
1603 17046 C3           CON(2) =pTRFMx
1604 17048 118          C=R0                   Recall hdlr address or src type
1605 1704B 831          ?XM=0                  Handled?
1606 1704E F3           GOYES  TFHD60
```

```
1607 17050 111          A=R1                 A(A) =  Dest file type
1608 17053 D5           B=C    A             B(A) =  Src file type
1609 17055 AF2          C=0    W             Set "TEXT to BASIC" copy code
1610 17058 8B0          ?B<A   A             Source < dest ?
1611 1705B 70           GOYES  TFHD40
1612 1705D B46          C=C+1  S             Clear "BASIC to TEXT" copy code
1613 17060 DC           ABEX   A             Reverse order, B(A) = lesser type
1614 17062 E6   TFHD40  C=C+1  A             Low type not TEXT?
1615 17064 8A5          ?B#C   A              .
1616 17067 00           RTNYES
1617 17069 3341         LC(4)  =fBASIC       High type not BASIC?
           2E
1618 1706F 8A6          ?A#C   A              .
1619 17072 00           RTNYES
1620 17074 34DD         LC(5)  TXT->B        Load TEXT->BASIC address
           671
1621 1707B 94A          ?C=0   S             Are we TEXT to BASIC?
1622 1707E F0           GOYES  TFHD60
1623 17080 A46          C=C+C  S             Set copy code = 4
1624 17083 A46          C=C+C  S              .
1625 17086 34E7         LC(5)  B->TXT        Load BASIC->TEXT address
           871
1626 1708D 860  TFHD60  ?ST=0  sTFREQ        Set carry if handler not required
1627 17090 00           RTNYES
1628 17092 03           RTNCC
1629
1630            ************************************************************
1631            *
1632            *  TFLINE  -  Read and Transform Line by Calling Handler
1633            *
1634            *  Entry:
1635            *       DO     @  /NUMLN
1636            *       /TFUAD =  Transform handler address
1637            *
1638            *  Exit:
1639            *       /STAT  =  Entry status (not restored)
1640            *       /SFIB# =  Source file FIB#
1641            *       Also see Transform handler exit conditions
1642            *
1643            *  TRANSFORM HANDLER INTERFACE:
1644            *
1645            *  Entry:
1646            *       P      =  0
1647            *       R4(15-14) = FIB#
1648            *       (INBS) = (OUTBS) = (AVMEMS)
1649            *       100 nibs of memory are available PLUS leeway
1650            *           during an inverse transformation
1651            *       300 nibs of memory are available PLUS leeway
1652            *           during a normal transformation
1653            *
1654            *  Exit:
1655            *       P      =  0
1656            *       sEOF   =  Set if input line read was the end of file
1657            *      Carry clear:
1658            *       /LNLEN =  Length of input line
```

```
1659                *       /LINE# =  5 digit BCD line number of input line
1660                *                 or, if 0, the current line count will
1661                *                 be used (this is used for TFM warnings)
1662                *       Output buffer contains output line
1663                *      Carry set:
1664                *       C(3-0) =  Error code
1665                *       C(S)   =  0 if non-recoverable error, else #0
1666                *
1667 17094 7092  TFLINE GOSUB  TFUSTS          Save status
1668 17098 7780         GOSUB  TFURDC          Read src FIB, collapse buffers
1669 1709C D2           C=0    A
1670 1709E 3146         LC(2)  100             Load mem margin for inverse tfm
1671 170A2 876          ?ST=1  sTFINV          In inverse transformation?
1672 170A5 70           GOYES  TFLI20
1673 170A7 32C2         LC(3)  300             Load up normal tfm mem margin
           1
1674 170AC 8F00  TFLI20 GOSBVL =MEMCKL         Not enough memory?
           000
1675 170B3 AC2          C=0    S                .
1676 170B6 400          RTNC                    . return if not
1677 170B9 164          DO=DO+ (/LINE#)-(/NUMLN)  Zero out line number
1678 170BC D0           A=0    A                .
1679 170BE 140          DATO=A A                .
1680 170C1 7C0F         GOSUB  TFURHA          Read handler address
1681 170C5 D6           C=A    A               Call routine
1682                ********************************
1683                * NOTE: FALLS INTO CODE BELOW  *
1684                ********************************
1685                ************************************************************
1686                **
1687                ** Name:    GOSUBI  -  Indirect GOSUB Utility
1688                **
1689                ** Purpose:
1690                **      Given an address, jump to it with carry preserved.
1691                **
1692                ** Entry:
1693                **      C(A)   =  Address to jump to
1694                **
1695                ** Exit:  To specified address
1696                **
1697                ************************************************************
1698                ********************************
1699                * NOTE: FALLEN INTO FROM ABOVE *
1700                ********************************
1701 170C7 06    GOSUBI RSTK=C                Push address onto stack
1702 170C9 01           RTN                    Go to it with carry preserved
1703
```

```
1704             CARYSV  STITLE Save Carry in C(S)
1705             ******************************************************************
1706             ******************************************************************
1707             **
1708             ** Name:    CARYSV  -  Save Carry in C(S)
1709             **
1710             ** Category:   GENUTL
1711             **
1712             ** Purpose:
1713             **      Save current setting of carry bit in C(S).  Carry can
1714             **      then be restored by:   C=C-1   S
1715             **
1716             ** Entry:
1717             **      None
1718             **
1719             ** Exit:
1720             **      Carry  =  Entry state
1721             **      C(S)   =  0  if carry is set
1722             **             =  1  if carry is clear
1723             **      C(A)   =  0  if carry clear, else entry state
1724             **
1725             ** Calls:      None.
1726             **
1727             ** Uses.......
1728             **  Inclusive: C(S)
1729             **
1730             ** Stk lvls:   0
1731             **
1732             ** History:
1733             **
1734             **    Date     Programmer              Modification
1735             **  --------   ----------   ---------------------------------
1736             **  10/11/82      FH        Designed and coded.
1737             **
1738             ******************************************************************
1739             ******************************************************************
1740 170CB AC2  =CARYSV C=0    S              Set C(S)
1741 170CE 400         RTNC                  Return if carry set
1742 170D1 B46         C=C+1  S              Increment C(S)
1743 170D4 D2          C=0    A              Clear C(A)
1744 170D6 03          RTNCC
```

```
1745                DE-REF  STITLE Remove File From Memory References
1746                ********************************************************************
1747                ********************************************************************
1748                **
1749                ** Name:    DE-REF  -  Remove File From Memory References
1750                **
1751                ** Category:   FILUTL
1752                **
1753                ** Purpose:
1754                **      Zero all pointers in I/O Buffers or GOSUB or FOR/NEXT
1755                **      stacks which point to the specified file.  Technique
1756                **      used is to call RFADJ with simulated purge of all but
1757                **      the last nib of the file (full file cannot be used or
1758                **      else pointers to the next file, such as CURRST, will
1759                **      get adjusted incorrectly).
1760                **
1761                ** Entry:
1762                **      P      =  0
1763                **      A(A)   =  Address of file header of file
1764                **
1765                ** Exit:
1766                **      P      =  0
1767                **      A(A)   =  Address of file header of file
1768                **      Carry  =  Clear
1769                **
1770                ** Calls:      FILSK+, RFAD-I, PUGFIB
1771                **
1772                ** Uses.......
1773                **  Exclusive: A(A), B(A), C(A), D1, R0, S-R0-1
1774                **  Inclusive: A-D,D0,D1,R0,R1
1775                **
1776                ** Stk lvls:   4
1777                **
1778                ** History:
1779                **
1780                **    Date     Programmer             Modification
1781                **  --------   ----------   ---------------------------------
1782                **  09/27/82      FH        Implemented check for current file
1783                **  03/12/83      FH        Revised to use simulated purge
1784                **
1785                ********************************************************************
1786                ********************************************************************
1787 170D8 D8    =DE-REF B=A    A              B = DEST = file header address
1788 170DA 8F00          GOSBVL =FILSK+        Compute C(A) = addr of next file
            000
1789 170E1 CE            C=C-1  A              Compute EOF-1 = last nib of file
1790 170E3 108           R0=C                  R0 = BEGIN SOURCE = last nib
1791 170E6 1F67          D1=(5) =S-R0-1        Save pointer to "End of Chain"
            8F2
1792 170ED 145           DAT1=C A              Save EOF-1 as End of Chain
1793 170F0 E1            B=B-C  A              B = OFFSET = DEST - SOURCE
1794 170F2 8F00          GOSBVL =RFAD-I        Adjust references
            000
1795 170F9 8C00          GOLONG =PUGFIB        Purge FIB if file was open
            00
```

```
1796            EOFCHK  STITLE Check for EOF on File
1797            ********************************************************************
1798            ********************************************************************
1799            **
1800            ** Name:    EOFCHK  -  Check for EOF on File
1801            **
1802            ** Category:   FILUTL
1803            **
1804            ** Purpose:
1805            **      Check if FIB's current position >= length of file.
1806            **
1807            ** Entry:
1808            **      STMTD1 =  Address of FIB entry
1809            **
1810            ** Exit:
1811            **      P      =  0
1812            **      D1     @  Data Len field of FIB
1813            **      STMTD1 =  Address of FIB entry
1814            **      A(A)   =  Contents of FIB Data Len field
1815            **      C(A)   =  Contents of FIB Curr Pos field
1816            **      S7     =  Set iff Curr Pos >= Data Len (sEOF)
1817            **      Carry  =  Set iff Curr Pos >= Data Len
1818            **
1819            ** Calls:      None.
1820            **
1821            ** Uses.......
1822            **  Inclusive: A(A),C(A),D1,P,S7(sEOF)
1823            **
1824            ** Stk lvls:   1
1825            **
1826            ** History:
1827            **
1828            **    Date     Programmer             Modification
1829            **  --------   ----------   ----------------------------------
1830            **  03/08/83      FH        Designed and coded.
1831            **
1832            ********************************************************************
1833            ********************************************************************
1834 170FF 74F0 =EOFCHK GOSUB  D1@CPS          Read curr pos field into C
1835 17103 175          D1=D1+ (oDLENb)-(oCPOSb)  Read data len into A
1836 17106 143          A=DAT1 A                    .
1837 17109 857          ST=1   =sEOF           Presume EOF
1838 1710C 8BA          ?A<=C  A                . At EOF?
1839 1710F 00           RTNYES                  .
1840 17111 847          ST=0   =sEOF           Flag no EOF
1841 17114 03           RTNCC
```

```
1842            FIB#SV  STITLE Save File FIB # in R4
1843            ********************************************************************
1844            ********************************************************************
1845            **
1846            ** Name:    FIB#SV  -  Store/Recall File FIB # in R4
1847            ** Name:    FIB#RS  -  Store/Recall File FIB # in R4
1848            **
1849            ** Category:   FILUTL
1850            **
1851            ** Purpose:
1852            **      Save and recall file FIB# into/from R4(15-14).
1853            **
1854            ** Entry:
1855            **    FIB#SV:
1856            **      A(B)      =  FIB#
1857            **    FIB#RS:
1858            **      R4(15-14) =  FIB#
1859            **
1860            ** Exit:
1861            **      A(B)      =  FIB#
1862            **      R4(15-14) =  FIB#
1863            **      Carry preserved
1864            **
1865            ** Calls:      None
1866            **
1867            ** Uses.......
1868            **  Inclusive: R4 (FIB#SV only), A (FIB#RS only)
1869            **
1870            ** Stk lvls:   0
1871            **
1872            ** History:
1873            **
1874            **    Date     Programmer              Modification
1875            **  --------   ----------   ----------------------------------
1876            **  09/29/82      FH        Designed and coded
1877            **
1878            ********************************************************************
1879            ********************************************************************
1880
1881            ********************************************************************
1882            *
1883            *  TFURDS, TFURDD, TFURD+  -  Read Source/Dest FIB#
1884            *
1885            *  Entry:
1886            *       None.
1887            *
1888            *  Exit:
1889            *       R4(15-14) = FIB#
1890            *       A(B)   =  Source FIB# (TFURDS)
1891            *              =  Dest   FIB# (TFURDD,TFURD+)
1892            *       D1     @  /SFIB#      (TFURDS)
1893            *                             (TFURDD,TFURD+)
1894            *       C(A)   =  0           (TFURD+)
1895            *
1896            *  Uses:
```

```
1897            *       A(B),D1,R4, C(A) (TFURD+)
1898            *
1899 17116 D2    TFURD+ C=0    A
1900 17118 1FBC  TFURDD D1=(5) /DFIB#
           8F2
1901 1711F 6110         GOTO   TFURD-
1902            *
1903 17123 8F00  TFURDC GOSBVL =BBCOLL        Collapse input, output buffers
           000
1904            *
1905 1712A 1F9C  TFURDS D1=(5) /SFIB#         Set up Source FIB read
           8F2
1906 17131 14B   TFURD- A=DAT1 B              Read FIB#
1907            *
1908 17134 814  =FIB#SV ASRC                  Store FIB#
1909 17137 814          ASRC                   .
1910 1713A 104          R4=A                   .
1911 1713D 114  =FIB#RS A=R4                  Restore FIB#
1912 17140 810          ASLC                   .
1913 17143 810          ASLC                   .
1914 17146 01           RTN
```

```
1915            HEXASC  STITLE Hexadecimal to Ascii Conversion
1916            ************************************************************
1917            ************************************************************
1918            **
1919            ** Name:(S) HEXASC  -  Convert Hexadecimal to Ascii
1920            **
1921            ** Category:   CONVRT
1922            **
1923            ** Purpose:  Converts specified number of hex digits to ASCII
1924            **           and returns the string (backwards) in A(W), B(W)
1925            **
1926            ** Entry:
1927            **      A(W)   =  Hex digits
1928            **      C(S)   =  #nibs-1 to convert; must be 7 or less
1929            **      P      =  0
1930            **
1931            ** Exit:
1932            **      A(W)   =  Converted string (high digit in low memory)
1933            **      B(W)   =  Converted string (high digit in low memory)
1934            **      C(S)   =  F
1935            **      P      =  0
1936            **      Carry  =  Set
1937            **
1938            ** Calls:    none
1939            **
1940            ** Stack lvls: 0
1941            **
1942            ** Uses:     A, B, C(S), C(B)
1943            **
1944            ** History:
1945            **
1946            **    Date       Programmer            Modification
1947            **  --------     ----------      ------------------------------
1948            **  07/04/82        SW           Added documentation
1949            **
1950            ************************************************************
1951            ************************************************************
1952            *
1953 17148 BF1  =HEXASC BSL    W
1954 1714B BF1          BSL    W
1955 1714E A88          B=A    P
1956 17151 BF4          ASR    W
1957 17154 30A          LCHEX  A
1958 17157 989          ?B>=C  P
1959 1715A 90           GOYES  HXASC3
1960 1715C 3103         LCHEX  30
1961 17160 560          GONC   HXASC5          (B.E.T.)
1962            *
1963 17163 3173  HXASC3 LCHEX  37
1964 17167 A61   HXASC5 B=B+C  B
1965 1716A A4E          C=C-1  S
1966 1716D 5AD          GONC   HEXASC
1967            *
1968 17170 AF4          A=B    W
1969 17173 01           RTN
```

```
1970                LIF>NB  STITLE Convert LIF1 Byte Count to Nibs
1971                ********************************************************************
1972                ********************************************************************
1973                **
1974                ** Name:    LIF>NB  -  Convert Byte Count to Nibs
1975                **
1976                ** Category:   FILUTL
1977                **
1978                ** Purpose:
1979                **      Takes the byte count from the header of a line from
1980                **      a TEXT (LIF type 1) file and converts it into the true
1981                **      number of nibs occupied by the remainder of the line.
1982                **      The trick is that if the byte count is odd, and extra
1983                **      byte of padding is added to the line.
1984                **
1985                ** Entry:
1986                **      A(3-0) =  Byte count from line header as a 16-bit value
1987                **                (note: the 4-nib value read from the file
1988                **                be byte reversed using SWPBYT beforehand).
1989                **
1990                ** Exit:
1991                **      C(A)   =  Corresponding nibble count
1992                **     Carry clear:   (Byte length was EVEN)
1993                **      S8     =  Clear (sODD)
1994                **     Carry set:     (Byte length was ODD)
1995                **      S8     =  Set   (sODD)
1996                **
1997                ** Calls:      None.
1998                **
1999                ** Uses.......
2000                **  Inclusive: C(A), S8(sODD)
2001                **
2002                ** Stk lvls:   0
2003                **
2004                ** History:
2005                **
2006                **    Date      Programmer              Modification
2007                **  --------    ----------   ---------------------------------
2008                **  09/21/82      FH         Designed and coded
2009                **
2010                ********************************************************************
2011                ********************************************************************
2012                *^^
2013                *
2014                * sODD must be greater than 4 because the HPIL ROM destroys S4
2015                * thru S0.  S8 is OK for sODD because sODD is only used for
2016                * READING a file, and HPIL I/O doesn't change S8 except while
2017                * WRITING a file to tape.       -NZ
2018                *
2019
2020                 sODD   EQU    8              (Modified by NZ on 8/16/83-was 4)
2021                *^^
2022
2023 17175 D6   =LIF>NB C=A    A              Set C(A) to bytes+1
2024 17177 E6          C=C+1  A               .
```

```
2025 17179 F2          CSL    A              .
2026 1717B F6          CSR    A              .
2027 1717D C6          C=C+C  A              Convert to nibs
2028 1717F 0B          CSTEX                 Set oddness status
2029 17181 861         ?ST=0  1               . set carry if odd bytes
2030 17184 50          GOYES  LIF>20
2031 17186 841         ST=0   1              Subtract extra byte
2032 17189 0B    LIF>20 CSTEX
2033 1718B 858         ST=1   sODD           Return status if odd
2034 1718E 400         RTNC
2035 17191 848         ST=0   sODD           Else return even status
2036 17194 03          RTNCC
```

```
2037            LOCFHD  STITLE Locate File Header Given FIB#
2038            ********************************************************************
2039            ********************************************************************
2040            **
2041            ** Name:     LOCFHD  -  Locate File Header Given FIB#
2042            ** Name:     LOCFH*  -  Locate File Header Given FIB Address
2043            ** Name:     LOCFH-  -  Locate File Header Given FIB Address
2044            **
2045            ** Category:   FILUTL
2046            **
2047            ** Purpose:
2048            **      Locates file header given FIB# of memory file
2049            **
2050            ** Entry:
2051            **    LOCFHD:
2052            **      R4(15-14)  =  File # in FIB
2053            **    LOCFH*:
2054            **      B(A)   =  FIB address
2055            **    LOCFH-:
2056            **      A(A)   =  FIB address
2057            **
2058            ** Exit:
2059            **      A(A)   =  File header address
2060            **      B(A)   =  FIB address
2061            **      D0     =  File header address
2062            **      R4(15-14)  =  File # in FIB
2063            **      Carry  =  Clear
2064            **      P      =  0 (set by LOCFHD only)
2065            **
2066            ** Calls:      LOCFIL (LOCFHD only)
2067            **
2068            ** Uses.......
2069            **  Exclusive: A(A), D1
2070            **  Inclusive: LOCFHD: A,B,C,D,D1,P
2071            **             LOCFH*: A(A), D1
2072            **             LOCFH-: A(A), D1
2073            **
2074            ** Stk lvls:   3 (LOCFHD), 0 (LOCFH*,LOCFH-)
2075            **
2076            ** History:
2077            **
2078            **    Date     Programmer            Modification
2079            **  --------   ----------   ----------------------------------
2080            **  09/29/82      FH        Designed and coded
2081            **
2082            ********************************************************************
2083            ********************************************************************
2084 17196 709F  TFUSFH GOSUB  TFURDS          Read source FIB
2085 1719A 7B70 =LOCFHD GOSUB  LOCFI#          Locate file FIB
2086 1719E D4   =LOCFH* A=B    A               Position at file header field
2087 171A0 131   LOCFH- D1=A                    .
2088 171A3 17C          D1=D1+ =oFBEGb          .
2089 171A6 15B5         A=DAT1 =lFBEGb         Read file begin/header address
2090 171AA 130          D0=A                    . into D0
2091 171AD 03           RTNCC
```

```
2092            LOCFIL  STITLE Locate File With FIB
2093            ******************************************************************
2094            ******************************************************************
2095            **
2096            ** Name:(S) LOCFIL  -  Locate File With FIB
2097            ** Name:    LOCFI#  -  Locate File With FIB
2098            **
2099            ** Category:   FILUTL
2100            **
2101            ** Purpose:
2102            **      Find FIB for file given file number and return position
2103            **      information.
2104            **
2105            ** Entry:
2106            **    LOCFIL:
2107            **      A(B)   =  FIB file number (LOCFI+ will return it in R4)
2108            **    LOCFI#:
2109            **      R4(15,14)  =  FIB file number
2110            **
2111            ** Exit:
2112            **      P      =  0
2113            **      R4(15,14) = FIB file number (LOCFI+, LOCFI# only)
2114            **     Carry clear:  FIB entry found
2115            **      A(x-0) =  "Data Begin" field of FIB entry
2116            **       (S)   =  Protection nibble from FIB
2117            **      B(A)   =  Address of FIB entry
2118            **      C(A)   =  "Current Position" field of FIB entry
2119            **      D(S)   =  Device code
2120            **       (A)   =  D(X) = Dev addr if external device, rest 0
2121            **             =  D(B) = Port id if port, rest 0
2122            **             =  0 if MAIN
2123            **      D1     @  "Current Position" field of FIB entry
2124            **      S7     =  Set if current position is at EOF (sEOF)
2125            **      S10    =  Set if external device (sI/OBF)
2126            **      STMTD1 =  Address of File FIB
2127            **     Carry set:    Error encountered
2128            **      C(3-0) =  eFnFND if FIB entry not found
2129            **             =  eNtIMP if external device
2130            **
2131            ** Calls:      FFIB#
2132            **
2133            ** Uses.......
2134            **  Inclusive: A,B,C,D,D1,P,S7(sEOF),S10(sI/OBF)
2135            **
2136            ** Stk lvls:   2
2137            **
2138            ** History:
2139            **
2140            **     Date     Programmer            Modification
2141            **   --------   ----------   ---------------------------------
2142            **  06/07/82      FH        Designed and coded
2143            **
2144            ******************************************************************
2145            ******************************************************************
2146 171AF 813   svinf+ DSLC                  Shift device info into D(A)
```

```
2147 171B2 8D00  svinfo GOVLNG =SVINFO
           000
2148
2149 171B9 131   LOCF05 D1=A
2150 171BC D8            B=A    A              Save FIB address in B
2151 171BE 178           D1=D1+ =oPROTb        Read protection nibble
2152 171C1 1534          A=DAT1 S               .
2153 171C5 172           D1=D1+ (oDEVCb)-(oPROTb)  Read device code
2154 171C8 AF2           C=0    W               .
2155 171CB 1574          C=DAT1 S               .
2156 171CF AC7           D=C    S               . and copy into D for now
2157 171D2 17F           D1=D1+ (oDBEGb)-(oDEVCb)+7  Move to data begin fields
2158 171D5 84A           ST=0   sI/OBF          Presume not external device
2159 171D8 94A   LOCF10  ?C=0   S               MAIN?
2160 171DB 51            GOYES  LOCF20
2161 171DD 14F           C=DAT1 B               Read port address
2162 171E0 A47           D=D+D  S               NOT External device?
2163 171E3 5C0           GONC   LOCF20            GOYES
2164 171E6 85A           ST=1   sI/OBF          Flag external device
2165 171E9 1C2           D1=D1- 3               Read device address
2166 171EC 1573          C=DAT1 X               .
2167 171F0 AF7   LOCF20  D=C    W               Transfer into D
2168 171F3 780F          GOSUB  EOFCHK          Flag EOF
2169               *
2170 171F7 1F69  D1@CPS  D1=(5) =STMTD1         Position to FIB entry
           8F2
2171 171FE 143           A=DAT1 A               .
2172 17201 131           D1=A                   .
2173 17204 17F           D1=D1+ 16              Read data begin
2174 17207 174           D1=D1+ (oDBEGb)-16     .
2175 1720A 15BA          A=DAT1 =lDBEGb         .
2176 1720E 17F           D1=D1+ 16              Read current position
2177 17211 172           D1=D1+ (oCPOSb)-(oDBEGb)-16
2178 17214 147           C=DAT1 A
2179 17217 03     LOCFcc RTNCC
2180               *
2181 17219 702F =LOCFI# GOSUB  FIB#RS          Restore FIB# from R4
2182               *
2183 1721D 8E00 =LOCFIL GOSUBL =FFIB#          Find file
           00
2184 17223 133           AD1EX                  Save pointer to FIB entry
2185 17226 1F69          D1=(5) =STMTD1         .
           8F2
2186 1722D 141           DAT1=A A               .
2187 17230 588           GONC   LOCF05
2188 17233 3300          LC(4)  =eFnFND         Return if file FIB not found
           00
2189 17239 02            RTNSC
```

```
2190                OVERCK  STITLE Check if Read or Write Overflows File
2191                ********************************************************************
2192                ********************************************************************
2193                **
2194                ** Name:    OVERCK  -  Check if Read or Write Overflows File
2195                **
2196                ** Category:   FILUTL
2197                **
2198                ** Purpose:
2199                **      Check if read or write overflows file.
2200                **
2201                ** Entry:
2202                **      STMTD1 = Address if FIB entry
2203                **      A(A)   =  Number of nibs to read or write
2204                **      S10    =  0 if internal file
2205                **
2206                ** Exit:
2207                **      B(A)   =  Number of nibs to read or write
2208                **      D1     @  Data length field of FIB
2209                **      R3     =  Number of nibs to read or write
2210                **      R0     = Abs and relative curr pos from GTPTRX
2211                **      STMTD1 = Address if FIB entry
2212                **     Carry clear:
2213                **      C(A)   =  Contents of current position of FIB
2214                **     Carry set:
2215                **      C(3-0) =  Error code: eEOFIL
2216                **
2217                ** Calls:      EOFCHK
2218                **
2219                ** Uses.......
2220                **  Inclusive: A,B,C,D0,D1,P,R0,R3,S7,S10,S11
2221                **
2222                ** Stk lvls:   2
2223                **
2224                ** History:
2225                **
2226                **    Date     Programmer              Modification
2227                **  --------   ----------   ----------------------------------
2228                **  03/08/83      FH        Designed and coded.
2229                **
2230                ********************************************************************
2231                ********************************************************************
2232 1723B 103   =OVERCK R3=A                    Save nib count
2233 1723E 8E00         GOSUBL =GTPTRX          Set R0 to abs, relative addresses
           00
2234 17244 113          A=R3                    B(A) = Nib count
2235 17247 D8           B=A    A                 .
2236 17249 72BE         GOSUB  EOFCHK           Get curr pos, data len, update sEOF
2237 1724D E0           A=A-B  A                Reduce data len by nib count
2238 1724F 470          GOC    Eeofil           Branch if underflow
2239 17252 8BE          ?C<=A  A                If no overflow, return carry clear
2240 17255 2C           GOYES  LOCFcc
2241 17257 3300  Eeofil LC(4)  =eEOFIL          Premature EOF
           00
2242 1725D 02           RTNSC
```

```
2243                    STITLE Transfer Jumps
2244                ********************************************************************
2245                *
2246                *  Transfer Jumps
2247                *
2248                ********************************************************************
2249
2250 1725F 8D00  =d0=obs GOVLNG =D0=OBS
          000
2251 17266 8D00   oblcmp GOVLNG =OBLCMP
          000
2252 1726D 8D00   obprd  GOVLNG =OBPRD
          000
2253 17274 865    rdinfx ?ST=0  sTFINP          Read dest info: src if in-place
2254 17277 90             GOYES  rdinfd
2255 17279 843    rdinfs ST=0   =sDEST
2256 1727C 6600           GOTO   rdinfo
2257 17280 853    rdinfd ST=1   =sDEST
2258 17283 8D00   rdinfo GOVLNG =RDINFO
          000
2259 1728A 20     Rstk<r P=     0               Restore only one address
2260 1728C 8C00 =rstk<r GOLONG =Rstk<R
          00
2261 17292 20     R<rstk P=     0               Save only one address
2262 17294 8C00 =r<rstk GOLONG =R<Rstk
          00
2263
2264 1729A DE     D0=/B+ ACEX   A
2265 1729C 1B68   D0=/BX D0=(5) =S-R1-1
          8F2
2266 172A3 142            A=DAT0 A
2267 172A6 01             RTN
2268
2269
```

```
2270            STATRS  STITLE Restore Status
2271            ********************************************************************
2272            ********************************************************************
2273            **
2274            ** Name:(S) STATRS  -  Restore Status
2275            ** Name:    STATR+  -  Restore Status
2276            **
2277            ** Category:   SAVUTL
2278            **
2279            ** Purpose:
2280            **      Restore status flags S11 - S0 and S13 from area saved
2281            **      by STATSV.  STATR+ merges specified bits from current
2282            **      status setting with restored S11 - S0.
2283            **
2284            ** Entry:
2285            **      D1      @  Save area written by STATSV
2286            **     STATR+:
2287            **      C(X)    =  Bits corresponding to status flags to be pre-
2288            **                 served from current status setting during
2289            **                 restore.
2290            **
2291            ** Exit:
2292            **      S13, S11 - S0 restored (merged w/input bits if STATR+)
2293            **      C(X)  =  Old S11 - S0
2294            **      Carry clear
2295            **
2296            ** Calls:      STATR+ calls STATRS which has no calls
2297            **
2298            ** Uses.......
2299            **  Inclusive: C(A), S13, S11-S0, A(A) for STATR+ only
2300            **
2301            ** Stk lvls:   0 (STATRS), 1 (STATR+)
2302            **
2303            ** History:
2304            **
2305            **    Date     Programmer              Modification
2306            **  --------   ----------   ---------------------------------
2307            **  06/15/82      FH        Designed and coded.
2308            **
2309            ********************************************************************
2310            ********************************************************************
2311
2312            ********************************************************************
2313            *
2314            *  TFUWRN
2315            *
2316            *  Entry:
2317            *       R0      =  Error code
2318            *
2319 172A8 7C70  TFUWRN GOSUB  TFUSTS         Save status
2320            ****************************************************
2321            *             ENTER STATUS-SAVED AREA                *
2322            ****************************************************
2323 172AC 84D          ST=0   13             Fake out MFWRN
2324 172AF 118          C=R0                  Recall error code
```

```
2325 172B2 8E00          GOSUBL =CSLC5          Position to secondary insertion
           00
2326 172B8 1D4E          D1=(2) =/LINE#         Line # to primary insertion
2327 172BC 147           C=DAT1 A                .
2328 172BF 2B            P=     #B               . straight bcd output
2329 172C1 8AE           ?C#0   A               Line # really there?
2330 172C4 B0            GOYES  TFUW20
2331 172C6 1DFD          D1=(2) /NUMLN          Supply line count instead
2332 172CA 147           C=DAT1 A                .
2333 172CD 2F            P=     #F               . hex to dec
2334 172CF 10A   TFUW20  R2=C                   Save insertion in R2
2335 172D2 80FE          CPEX   14              Insert format specification
2336 172D6 2D            P=     13              Specify 5 digits
2337 172D8 304           LC(1)  5-1              .
2338 172DB 20            P=     0
2339 172DD 3300          LC(4)  =eTFWRN         Load Transform warning hdr
           00
2340 172E3 2E            P=     #E              Issue warning
2341 172E5 8F00          GOSBVL =MFWRNQ          .
           000
2342             ************************************************************
2343             *  FALL INTO FOLLOWING CODE TO LEAVE STATUS-SAVED AREA  *
2344             ************************************************************
2345
2346             ***
2347             *  Name:  TFUSTR  -  Restore Status From TRFMBF Area
2348             *
2349 172EC 1F6D  TFUSTR D1=(5) /STAT
           8F2
2350             *
2351 172F3 147  =STATRS C=DAT1 A               Read save area
2352 172F6 0B           CSTEX                  Restore S11 - S0
2353 172F8 816          CSRC                   Test old S13 (set if C(3) = 1)
2354 172FB A2E          C=C-1  XS               .
2355 172FE 812          CSLC                    .
2356 17301 85D          ST=1   13              Presume S13 set
2357 17304 500          RTNNC
2358 17307 84D          ST=0   13              Clear S13
2359 1730A 03    STATcc RTNCC
2360             *
2361 1730C DA   =STATR+ A=C    A               Save merge bits
2362 1730E 71EF         GOSUB  STATRS          Recall status
2363 17312 0EF2         C=A&C  A               Pick out preservation bits
2364 17316 0B           CSTEX                  Pick out restoration bits
2365 17318 FC           A=-A-1 A                .
2366 1731A 0EF6         A=A&C  A                .
2367 1731E 0B           CSTEX                  Merge
2368 17320 0EFA         C=A!C  A                .
2369 17324 0B           CSTEX                  Merge bits
2370 17326 03           RTNCC
```

```
2371            STATSV  STITLE Save Status
2372            ********************************************************************
2373            ********************************************************************
2374            **
2375            ** Name:(S) STATSV  -  Save Status S13, S11 - S0
2376            **
2377            ** Category:   SAVUTL
2378            **
2379            ** Purpose:
2380            **      Save status flags S13, S11 - S0 in designated spot.
2381            **
2382            ** Entry:
2383            **      D1      @  Start of 4-nib save area
2384            **
2385            ** Exit:
2386            **      Save area written (see detail below)
2387            **      Carry clear
2388            **
2389            ** Calls:      None
2390            **
2391            ** Uses.......
2392            **  Exclusive: C(A)
2393            **  Inclusive: C(A)
2394            **
2395            ** Stk lvls:   0
2396            **
2397            ** Detail:
2398            **      Save area:   Nibs          Contents
2399            **                   ----          --------
2400            **                   2-0           Status S11 - S0
2401            **                    3            0 is S13 clear, 1 if set
2402            **
2403            ** History:
2404            **
2405            **    Date      Programmer              Modification
2406            **  --------   ----------   ----------------------------------
2407            **  06/15/82      FH        Designed and coded.
2408            **
2409            ********************************************************************
2410            ********************************************************************
2411 17328 1F6D  TFUSTS D1=(5) /STAT
           8F2
2412 1732F D2   =STATSV C=0    A             Save S13
2413 17331 86D          ?ST=0  13             .
2414 17334 50           GOYES  STAT40         .
2415 17336 B26          C=C+1  XS             .
2416 17339 F2    STAT40 CSL    A             Position S13 nib
2417 1733B 09           C=ST                 Save S11 - S0
2418 1733D 15D3         DAT1=C 4             Write to save area
2419 17341 03           RTNCC
```

```
2420            PURGEF  STITLE Purge Internal or External File
2421            ************************************************************
2422            ************************************************************
2423            **
2424            ** Name:(S) PURGEF  -  Purge Internal or External File
2425            **
2426            ** Category:   FILUTL
2427            **
2428            ** Purpose:
2429            **      Purge file given its FSPECx information.
2430            **
2431            ** Entry:
2432            **      P       =  0
2433            **      A(W)    =  First 8 chars of file name.
2434            **      R0(3-0)=  Last 2 chars of file name.
2435            **      D(S)    =  Device code
2436            **      D(3-0)  =  Secondary device info
2437            **
2438            ** Exit:
2439            **      P       =  0
2440            **      File purged.  If file not found, error ignored.
2441            **
2442            ** Calls:
2443            **      FINDF, PRGFMF, POLL
2444            **
2445            ** Uses.......
2446            **  Inclusive: A-D,D0,D1,P,R0,R1,S-R0-0,S-R0-1,S7,S8
2447            **      If purging current file: also R2,R3,S9,S10,S11,S7-S0
2448            **        "  "  "  LEX      "    also R2,R3,S9
2449            **
2450            ** Stk lvls:   6
2451            **
2452            ** History:
2453            **
2454            **     Date    Programmer             Modification
2455            **   --------  ----------   ---------------------------------
2456            **  06/07/82      FH        Designed and coded
2457            **  06/09/83      FH        Expanded to include external files
2458            **
2459            ************************************************************
2460            ************************************************************
2461 17343 71EF  TFUPGD GOSUB  TFUSTS        Save status
2462 17347 792F        GOSUB  rdinfx        Recall source/dest file info
2463 1734B 817         DSRC                  .
2464 1734E A06         C=C+C  P
2465 17351 7400        GOSUB  PURGEF
2466 17355 6EAC        GOTO   TFUSVE        Restore status, preserve error msg
2467
2468 17359 ACB  =PURGEF C=D    S             Test for external file
2469 1735C A46         C=C+C  S              .
2470 1735F 441         GOC    PURG20         .
2471 17362 8E00        GOSUBL =findf         Find file
            00
2472 17368 D2          C=0    A              . and clear out with null error
2473 1736A 400         RTNC                  . if not there
```

```
2474 1736D 8D00          GOVLNG =PRGFMF        Purge file
           000
2475               *
2476 17374 8E00  PURG20  GOSUBL =POLL          Poll to purge external file
           00
2477 1737A 01            CON(2) =pPURGE
2478 1737C 01            RTN
```

```
2479            ?PRFIL  STITLE Check Protection
2480            ********************************************************************
2481            ********************************************************************
2482            **
2483            ** Name:(S) ?PRFIL  -  Check File Protection
2484            ** Name:(S) ?PRFI+  -  Check File Protection
2485            **
2486            ** Category:   FILUTL
2487            **
2488            ** Purpose:
2489            **      Checks file protection nib returned by LOCFIL for
2490            **      privacy (?PRFIL) or security (?PRFI+).
2491            **
2492            ** Entry:
2493            **      P      =  0
2494            **      A(S)   =  Protection nibble
2495            **
2496            ** Exit:
2497            **      P      =  0
2498            **     Carry set:
2499            **      C(3-0) =  File protection error code (eFPROT).
2500            **
2501            ** Calls:      None.
2502            **
2503            ** Uses.......
2504            **  Inclusive: C(S), C(3-0)
2505            **
2506            ** Stk lvls:   0
2507            **
2508            ** History:
2509            **
2510            **    Date      Programmer              Modification
2511            **  --------    ----------   ----------------------------------
2512            **  08/24/82      FH         Designed and coded
2513            **
2514            ********************************************************************
2515            ********************************************************************
2516 1737E 0C   =?PRFIL P=P+1                  Set P=2 for privacy
2517 17380 0C   =?PRFI+ P=P+1                  Set P=1 for security
2518 17382 80FF          CPEX   15             Position mask
2519 17386 20            P=     0               .
2520 17388 0E42          C=A&C  S              Test protection bits
2521 1738C 3300          LC(4)  =eFPROT        Load error code
           00
2522 17392 94E           ?C#0   S              Protection bits set?
2523 17395 00            RTNYES
2524 17397 03    ?PRFcc  RTNCC                 Return no error
```

```
2525            RRADD   STITLE Read Relocatable Address
2526            ********************************************************************
2527            ********************************************************************
2528            **
2529            ** Name:    RRADD   -  Read Relative Address From Table
2530            **
2531            ** Category:   GENUTL
2532            **
2533            ** Purpose:
2534            **      Read relative address and computes its absolute value.
2535            **
2536            ** Entry:
2537            **      D1      @  Table entry (5 nibs)
2538            **
2539            ** Exit:
2540            **      D1      @  After table entry
2541            **      C(A)    =  Absolute address if nonzero table entry
2542            **              =  0  if table entry was zero
2543            **      Carry   =  Clear if nonzero table entry
2544            **              =  Set if table entry was zero
2545            **
2546            ** Calls:      None
2547            **
2548            ** Uses.......
2549            **  Inclusive: A(A), C(A), D1
2550            **
2551            ** Stk lvls:   None
2552            **
2553            ********************************************************************
2554            ********************************************************************
2555
2556            ********************************************************************
2557            *
2558            *  TFURWD,TFURWS  -  Rewind (Source) File, Read Inverse Tfm Add
2559            *
2560            *  Entry:
2561            *     RFURWS:
2562            *       None:
2563            *     RFURWD:
2564            *       R4(15-14) = FIB#
2565            *
2566            *  Exit:
2567            *       D1      =  Transform handler address
2568            *       C(A)    =  Inverse tfm handler address
2569            *       Carry   =  Set if no inverse tfm address
2570            *
2571 17399 7D8D  TFURWS GOSUB  TFURDS          Read source FIB#
2572 1739D 778F  TFURWD GOSUB  TFUSTS          Save status
2573 173A1 747E         GOSUB  LOCFI#          Locate file
2574 173A5 D4           A=B    A               Position FIB entry address
2575 173A7 8E00         GOSUBL =REWIND         Rewind file
          00
2576 173AD 7B3F         GOSUB  TFUSTR          Restore status
2577
2578            ************************************
```

```
2579                *  NOTE: FALL INTO FOLLOWING CODE  *
2580                ***********************************
2581
2582                ********************************************************************
2583                *
2584                *  TFURIA  -  Read Inverse Transform Handler Address
2585                *
2586                *  Entry:
2587                *       None.
2588                *
2589                *  Exit:
2590                *       D1      =  Transform handler address
2591                *      Carry clear:  [Inverse transform address present]
2592                *       C(A)   =  Absolute address of inverse transform handler
2593                *      Carry set:    [Inverse transform address NOT present]
2594                *       C(A)   =  0
2595                *
2596                *  Uses:
2597                *       A(A),C(A),D1
2598                *
2599                ************************************
2600                *  NOTE: FALLEN INTO FROM ABOVE    *
2601                ************************************
2602 173B1 7C1C  TFURIA GOSUB  TFURMA          Fetch transform address
2603 173B5 131          D1=A                  Fetch inverse address
2604 173B8 1C4          D1=D1- 5               .
2605                ************************************
2606                *  NOTE: FALLEN INTO FROM ABOVE    *
2607                ************************************
2608 173BB 147  =RRADD  C=DAT1 A               Read address
2609 173BE 133          AD1EX                 Fetch entry address
2610 173C1 131          D1=A                   .
2611 173C4 174          D1=D1+ 5              Bump entry address ptr
2612 173C7 8AA          ?C=0   A              Zero address?
2613 173CA 00           RTNYES
2614 173CC C2           C=C+A  A              Compute absolute address
2615 173CE 03           RTNCC
```

```
2616            TRCFI#  STITLE Truncate File
2617            ********************************************************************
2618            ********************************************************************
2619            **
2620            ** Name:    TRCFI#  -  Truncate File at Current Position
2621            **
2622            ** Category:   FILUTL
2623            **
2624            ** Purpose:
2625            **      Sets end of data to current position and, for internal
2626            **      files, collapses out any trailing portion of the file.
2627            **
2628            ** Entry:
2629            **      R4(15-14) = File # in FIB.
2630            **
2631            ** Exit:
2632            **      P       =  0
2633            **
2634            ** Calls:       LOCFI#, LOCFH-, FILSK+, MVMEM+
2635            **
2636            ** Uses.......
2637            **  Inclusive: A-D,D0,D1,R0-R2,P,S10,S7,SCRTCH(4-0)
2638            **
2639            ** Stk lvls:   3
2640            **
2641            ** History:
2642            **
2643            **    Date      Programmer              Modification
2644            **  --------    ----------   ----------------------------------
2645            **  09/29/82       FH        Designed and coded.
2646            **  03/16/83       FH        Combined with closef code.
2647            **
2648            ********************************************************************
2649            ********************************************************************
2650 173D0 754E =TRCFI# GOSUB  LOCFI#          Locate file
2651 173D4 400          RTNC                    .
2652 173D7 175          D1=D1+ (oDLENb)-(oCPOSb) Set end of data to current pos
2653 173DA 145          DAT1=C A                .
2654 173DD 87A          ?ST=1  sI/OBF           Return if external file
2655 173E0 00           RTNYES
2656 173E2 CA           A=A+C  A                B  = DEST = abs curr position
2657 173E4 DC           ABEX   A                   . and A = FIB addr
2658 173E6 76BD         GOSUB  LOCFH-           D0 = File header address
2659 173EA 8F00         GOSBVL =FILSK+          A  = SOURCE = Eof
             000
2660 173F1 DA           A=C    A                .
2661 173F3 E8           B=B-A  A                B  = OFFSET = DEST - SOURCE
2662 173F5 136          CDOEX                   C  = HEADER
2663 173F8 8D00         GOVLNG =MVMEM+          Move memory
             000
```

```
2664                RDBAS   STITLE Read Line From Basic File
2665                ********************************************************************
2666                ********************************************************************
2667                **
2668                ** Name:(S) RDBAS   -  Read Line From Basic File
2669                **
2670                ** Category:   FILUTL
2671                **
2672                ** Purpose:
2673                **      Read a line from a BASIC file given the file's FIB.
2674                **      For memory files, FIB is spaced past line but no data
2675                **      is copied to output buffer.  For external files, line
2676                **      read is copied to output buffer.
2677                **
2678                ** Entry:
2679                **      R4(15-14) = File FIB#
2680                **      OUTBS  @  Start of output buffer
2681                **      (AVMEMS)  =  (OUTBS)
2682                **
2683                ** Exit:
2684                **      P       =  0
2685                **     Carry clear:  Line read
2686                **      S7      =  Set if file was positioned at EOF at operation
2687                **                 start, hence no data read (sEOF)
2688                **      C(A)    =  Full len (nibs) of line in file counting line
2689                **                 header.  Zero if S7(sEOF) set
2690                **      R3      =  Pointer to start of data read (in file or in
2691                **                 output buffer) unless S7(sEOF) set.
2692                **     Carry set:
2693                **      C(3-0)  =  Error code:
2694                **
2695                ** Calls:      READNB, RECNIB, TFUEOF, EOLSN7, FIBUPD, LOCFI#
2696                **
2697                ** Uses.......
2698                **  Inclusive: A-D,D0,D1,R0-R3,STMTR1,STMTD1,S11-S9,S7,S6,S4-S0
2699                **
2700                ** Stk lvls:   5
2701                **
2702                ** History:
2703                **
2704                **    Date      Programmer              Modification
2705                **  --------    ----------   ----------------------------------
2706                **  12/15/82      FH         Designed and coded.
2707                **
2708                ********************************************************************
2709                ********************************************************************
2710 173FF 72AB =RDBAS  GOSUB  TFUEOF          Locate file, return if error or EOF
2711 17403 1B17         D0=(5) =STMTR0         Save current position
           8F2
2712 1740A 144          DAT0=C A                .
2713 1740D C2           C=C+A  A               R3 = Abs start
2714 1740F 10B          R3=C                    .
2715 17412 134          D0=C                   Space beyond line number
2716 17415 87A          ?ST=1  sI/OBF          External file?
2717 17418 73           GOYES  RDBA40
```

```
2718 1741A 161          DO=DO+ 2              . minus 2 for EOLSCN
2719 1741D 3100         LC(2)  =tEOL
2720 17421 8F00         GOSBVL =EOLSN7        Space to end of line
           000
2721 17428 161          DO=DO+ 2              Space over EOL token
2722 1742B 132          ADOEX                 Set A = len
2723 1742E 11B          C=R3                   .
2724 17431 EA           A=A-C  A               .
2725 17433 859          ST=1   sREAD          Update FIB currpos
2726 17436 7FA0         GOSUB  FIBUPD          .
2727              *
2728              *  Update FIB, return line length
2729              *
2730 1743A 7BDD  RDBA20 GOSUB  LOCFI#         Locate file
2731 1743E 1F17         D1=(5) =STMTR0        Recall previous position
           8F2
2732 17445 143          A=DAT1 A               .
2733 17448 E2           C=C-A  A              Return length
2734 1744A 847          ST=0   sEOF            .
2735 1744D 03    RDBAcc RTNCC
2736
2737 1744F 7CB0  RDBA40 GOSUB  READN4         Read line number
2738 17453 400          RTNC
2739 17456 77B0  RDBA60 GOSUB  READN2         Read offset
2740 1745A 400          RTNC                   .
2741 1745D 7D52         GOSUB  RECNIB          .
2742 17461 D2           C=0    A               .
2743 17463 AE6          C=A    B               . into C
2744 17466 7EA0         GOSUB  READNB         Read line segment
2745 1746A 400          RTNC
2746 1746D D2           C=0    A              Recall last byte
2747 1746F 302          LCHEX  2               .
2748 17472 7B42         GOSUB  RECAL-          .
2749 17476 3100         LC(2)  =tEOL          Not EOL token?
2750 1747A 966          ?A#C   B               .
2751 1747D 9D           GOYES  RDBA60
2752 1747F 7CDD         GOSUB  dO=obs         Set R3 = OUTBS
2753 17483 10B          R3=C                   .
2754 17486 53B          GONC   RDBA20         BET: Compute & return line length
```

```
2755                RDTEXT  STITLE Read Line From Text File
2756                ********************************************************************
2757                ********************************************************************
2758                **
2759                ** Name:(S) RDTEXT  -  Read Line From Text File
2760                **
2761                ** Category:   FILUTL
2762                **
2763                ** Purpose:
2764                **      Read a line from a text file into the output buffer
2765                **      given the file's FIB.  The line's length header or
2766                **      EOF mark are not copied into the output buffer.
2767                **
2768                ** Entry:
2769                **      R(15-14) =  File FIB#
2770                **      OUTBS  @  Start of output buffer
2771                **      AVMEMS @  (OUTBS)
2772                **
2773                ** Exit:
2774                **      P      =  0
2775                **      OUTBS  @  Start of output buffer.
2776                **      AVMEMS @  After last nib read.
2777                **     Carry clear:  Line read
2778                **      S7     =  Set if file positioned at EOF. (sEOF)
2779                **      C(A)   =  Full len (nibs) of line in file counting line
2780                **                header.  Zero if no EOF marker at end of file.
2781                **     Carry set:
2782                **      C(3-0) =  Error code:
2783                **
2784                ** Calls:      TFUEOF, READNB, RECNIB, SWPBYT, LIF>NB, OBPRD
2785                **
2786                ** Uses.......
2787                **  Inclusive: A-D,D0,D1,R0-R3,P,S11-S9,S7,S6,S4-S0
2788                **
2789                ** Stk lvls:   5 plus 1 RSTKBF level
2790                **
2791                ** History:
2792                **
2793                **    Date     Programmer             Modification
2794                **  --------   ----------   ----------------------------------
2795                **  06/12/82      FH        Designed and coded.
2796                **  09/21/82      FH        Revised to fix byte reversal in
2797                **                            line header
2798                **
2799                ********************************************************************
2800                ********************************************************************
2801 17489 781B =RDTEXT GOSUB  TFUEOF          Return if EOF or missing FIB
2802 1748D 7E70        GOSUB  READN4          Read line header to A
2803 17491 400         RTNC                    .
2804 17494 7622        GOSUB  RECNIB           .
2805 17498 145         DAT1=C A                . (restore old av mem start)
2806 1749B 857         ST=1   =sEOF           Preset EOF conditions
2807 1749E D2          C=0    A                .
2808 174A0 23          P=     3               Test for FFFF
2809 174A2 D8          B=A    A                .
```

```
2810 174A4 B15          B=B+1  WP              .
2811 174A7 442          GOC    RDTE20           . and exit if found
2812 174AA 7675         GOSUB  SWPBYT           Remove byte reversal
2813 174AE 73CC         GOSUB  LIF>NB           Convert to true nib line length
2814 174B2 7260         GOSUB  READNB           Read line
2815 174B6 400          RTNC                    .
2816 174B9 708D         GOSUB  obprd            Set D1@AVS
2817 174BD 11B          C=R3                    Set C = Nib count read
2818 174C0 868          ?ST=0  sODD             NO oddness?
2819 174C3 90           GOYES  RDTE20
2820 174C5 CC           A=A-1  A                Reduce AVMEMS by one byte
2821 174C7 CC           A=A-1  A                .
2822 174C9 141          DAT1=A A                .
2823 174CC 23    RDTE20 P=     4-1              Add nibs for line header
2824 174CE 809          C+P+1                   .
2825 174D1 20           P=     0                Return
2826 174D3 03           RTNCC
2827             *
2828 174D5 0            CON(1) =FIXSPC
2829 174D6              BSS    20-1
2830             *
```

```
2831            READNB  STITLE Read/Write Line From/To File
2832            ************************************************************
2833            ************************************************************
2834            **
2835            ** Name:(S) READNB  -  Read/Write Nibs To/From File
2836            ** Name:(S) WRITNB  -  Read/Write Nibs To/From File
2837            **
2838            ** Category:   FILUTL
2839            **
2840            ** Purpose:
2841            **      Write a line to a file given its FIB file number.  File
2842            **      may reside in memory or on external device.  File will
2843            **      be positioned to start of previous line before the line
2844            **      is written.
2845            **
2846            ** Entry:
2847            **      R4(15-14)  =  Number of file in FIB
2848            **      C(A)   =  #Nibs to read if reading
2849            **      R3(A)  =  Length of previous line in nibs if writing
2850            **                  into memory
2851            **      Output buffer contains line to write if writing
2852            **
2853            ** Exit:
2854            **      P      =  0
2855            **      R4(15-14) = FIB#
2856            **     Carry set:
2857            **      C(A)   =  Error code:
2858            **                  Insufficient Memory, etc.
2859            **                  End of file (file is not altered)
2860            **     Carry clear:
2861            **      R3     =  #Nibs read or written, or offset if writing to
2862            **                  memory.
2863            **      S7     =  Set iff file at EOF after operation (sEOF)
2864            **      FIB spaced past line in file
2865            **      Output buffer collapsed if writing
2866            ** Calls:      NIBLIO
2867            **
2868            ** Uses.......
2869            **  Inclusive: A-D,R0-R3,D0,D1,P,STMTR1,STMTD1
2870            **             S11-S9,S8(WRITNB only),S7,S6,S4-S0
2871            **
2872            ** Stk lvls:   4   plus 1 RSTKBF level
2873            **
2874            ** NOTE:
2875            **      NO CHECK IS MADE whether the file is protected or in ROM.
2876            **
2877            ** Algorithm:
2878            **
2879            ** History:
2880            **
2881            **     Date    Programmer              Modification
2882            **   --------  ----------   ---------------------------------
2883            **  06/15/82      FH        Designed and coded.
2884            **
2885            ************************************************************
```

```
2886            ****************************************************************
2887
2888             sODDST EQU    6              "Odd nibble start" flag
2889             sODDEN EQU    7              "Odd nibble end" flag
2890             sREAD  EQU    9              "Reading" flag
2891
2892             /NSTAT EQU    S-R1-2         NIBLIO status storage
2893            ***
2894            *
2895            *  FIBUPD - Update FIB currpos, and dlen (only if write)
2896            *
2897            *  Entry:
2898            *       A(A)   =  Offset to current position
2899            *       R4(15-14) = FIB#
2900            *
2901            *  Note:  Does NOT read in corresponding sector for
2902            *    external files; use UPCPOS and STFPTR instead.
2903            *
2904 174E9 102   FIBUPD R2=A
2905 174EC 792D        GOSUB  LOCFI#          Locate file
2906 174F0 112         A=R2
2907 174F3 C2          C=C+A  A               Update current position
2908 174F5 145         DAT1=C A                .
2909 174F8 879         ?ST=1  sREAD           READing?
2910 174FB 21          GOYES  FIBUcc
2911 174FD 87A         ?ST=1  sI/OBF          External file?
2912 17500 D0          GOYES  FIBUcc
2913 17502 175         D1=D1+ (oDLENb)-(oCPOSb)
2914 17505 147         C=DAT1 A               Update Data Len
2915 17508 C2          C=C+A  A                .
2916 1750A 145         DAT1=C A                .
2917 1750D 03    FIBUcc RTNCC
2918            *
2919 1750F 22    READN4 P=     2              Read 4 nibs
2920 17511 0C    READN2 P=P+1                 Read 2 nibs
2921 17513 D2    READWP C=0    A              Set up entry condition for READNB
2922 17515 809         C+P+1                   .
2923            *
2924 17518 859  =READNB ST=1   sREAD
2925 1751B 109         R1=C                   Save away nib count in R1
2926 1751E 7AD4        GOSUB  OBAPND           .
2927 17522 400         RTNC
2928 17525 136         CDOEX                  Set C = Buffer start = old (AVMEMS)
2929 17528 5C0         GONC   NIBLIO          BET
2930
2931 1752B 849  =WRITNB ST=0   sREAD
2932 1752E 743D        GOSUB  oblcmp          C  = (OUTBS)
2933 17532 101         R1=A                   R1 = Nib count
2934
2935            ****************************************************************
2936            ****************************************************************
2937            **
2938            ** Name:    NIBLIO  -  Read/Write Nibbles To/From File
2939            **
2940            ** Category:   LOCAL
```

```
2941                    **
2942                    ** Purpose:    Perform nibble-oriented read/writes on opened file
2943                    **      in memory or external device.
2944                    **
2945                    ** Entry:
2946                    **      C(A)   = Buffer start (OUTBS if writing; old AVMEMS if
2947                    **                 reading)
2948                    **      R4(15-14) = FIB#
2949                    **      R1     = Nib count
2950                    **      S9     = Set iff READNB (sREAD)
2951                    **      R3(A)  = Previous line length in nibs iff WRITNB
2952                    **
2953                    ** Exit:
2954                    **      R3(A)  = Nib count read/written, or offset if memory
2955                    **                    write
2956                    **      R4(15-14) = FIB#
2957                    **
2958                    **
2959                    **  Algorithm:
2960                    **      Save FIB#
2961                    **      If READing, then
2962                    **        Set R2 to source start = (AVMEMS)
2963                    **        Open enough room at avmems for nibs to be read
2964                    **        If memerr, return error
2965                    **      Else
2966                    **        Set R2 to target start = (OUTBS)
2967                    **        Set D(A) to nib count  = (AVMEMS)-(OUTBS)
2968                    **
2969                    **
2970                    ************************************************************
2971                    ************************************************************
2972                    *
2973                    **      If nib count is 0, then return
2974                    **      Fetch abs current position, updating sI/OBF
2975                    **      Locate file
2976                    **      If writing into memory, then
2977                    **        Replace line
2978                    **        Update FIB
2979                    **        Return
2980                    *
2981 17535 736D  NIBLIO GOSUB  DO=/BX          Save buffer start
2982 17539 144         DAT0=C A               . in  S-R1-1
2983 1753C 10A         R2=C                   . and R2
2984 1753F 7F4D        GOSUB  R<rstk          Save return address
2985 17543 72DC        GOSUB  LOCFI#          Locate file
2986 17547 4A2         GOC    RSTKR-          Error if FIB not found
2987 1754A C2          C=A+C  A               C = abs curr pos for mem file
2988 1754C 111         A=R1                   Recall nib count
2989 1754F 879         ?ST=1  sREAD           Not writing?
2990 17552 A3          GOYES  NIBL40
2991 17554 87A         ?ST=1  sI/OBF          External file?
2992 17557 F2          GOYES  NIBL35
2993 17559 714C        GOSUB  LOCFH*          Fetch file header address
2994 1755D DE          ACEX   A               A = CURR POS, C = FILE HEADER
2995 1755F 8F00        GOSBVL =RPLLIN         Replace nibs in file
```

```
            000
 2996 17566 4B0          GOC    RSTKR-
 2997 17569 113   NIBL30 A=R3                    Recall nib count/offset
 2998 1756C 797F         GOSUB  FIBUPD          Update FIB
 2999               *
 3000 17570 21    RSTKR+ P=     1                Set up for carry clear
 3001 17572 0D    RSTKR- P=P-1                   Set or clear carry
 3002 17574 735B  RSTKRT GOSUB  CARYSV          Save carry
 3003 17578 AFF          CDEX   W                Save any error code
 3004 1757B 7B0D         GOSUB  Rstk<r          Restore return address
 3005 1757F DF           CDEX   A                Restore error code
 3006 17581 A4F          D=D-1  S                Restore carry
 3007 17584 01           RTN
 3008               *
 3009               **     If nothing to read/write, then return
 3010               **     If current position >= EOF, then return error
 3011               **     If READing from memory, then
 3012               **       MOVE*M the data into target buffer
 3013               **       Update FIB
 3014               **       Return
 3015               *
 3016 17586 8E00  NIBL35 GOSUBL =SETWRT         Set write access on buffer
            00
 3017 1758C 8A8   NIBL40 ?A=0   A               Nothing to read/write?
 3018 1758F 1E           GOYES  RSTKR+
 3019 17591 76AC         GOSUB  OVERCK          Check for overflow, set up R0, R3:
 3020               *
 3021               *                              R0(15-14) = relative byte offset
 3022               *                                in current sector if ext file
 3023               *                              R0(A)     = abs curr pos address
 3024               *                              R3(A)     = nib count
 3025               *
 3026 17595 4CD          GOC    RSTKR-          Return if error
 3027 17598 87A          ?ST=1  sI/OBF          Not reading from memory?
 3028 1759B F0           GOYES  NIBL50
 3029 1759D 110          A=R0                   A = SOURCE
 3030 175A0 11A          C=R2                   C = TARGET = (OUTBS)
 3031 175A3 7011         GOSUB  move*m          Move data into output buffer
 3032 175A7 51C          GONC   NIBL30          BET: Update FIB
 3033               *
 3034               **     Set R1 to FIB's Record length field for UPCPOS later
 3035               **     If current position is odd, then
 3036               **       Set "Odd start" flag
 3037               **       Open up 1 extra nib at buffer start
 3038               **       Return error if memerr
 3039               *
 3040 175AA 1C9   NIBL50 D1=D1- (oDLENb)-(oRECLb) Set R1 = Record len (needed
 3041 175AD 143          A=DAT1 A                . DATA files only) for use
 3042 175B0 101          R1=A                    . by UPCPOS later
 3043 175B3 75EC         GOSUB  D0=/BX          A = SOURCE = BUFFER START
 3044 175B7 822          SB=0                   Test for odd nibble curr pos
 3045 175BA 846          ST=0   sODDST           . clear "Odd start" flag
 3046 175BD 81E          CSRB                    .
 3047 175C0 832          ?SB=0                  Was current position even?
 3048 175C3 01           GOYES  NIBL55           .
```

```
3049 175C5 856          ST=1   sODDST        Set "Odd start" flag
3050 175C8 D2           C=0    A             C = 1
3051 175CA E6           C=C+1  A              .
3052 175CC 77B0         GOSUB  OBEDIT        Open up extra nib at buffer st
3053 175D0 41A          GOC    RSTKR-        Return if error
3054              *
3055              **    Fetch source and destination pointers
3056              **    If current position is odd, then
3057              **      Increment working nib count
3058              **      If writing, then
3059              **        Read nib from abs curr pos-1 into av mem st
3060              **    Clear "Odd end" flag
3061              **    Compute working number of whole bytes to read or write
3062              **    If odd working nib count, then
3063              **      Set "Odd end" flag
3064              *
3065 175D3 131   NIBL55 D1=A                  D1 @ buffer start
3066 175D6 110          A=R0                 D0 @ abs curr pos (byte resolution)
3067 175D9 130          D0=A                  .
3068 175DC 113          A=R3                 Set B = working nib count
3069 175DF AF1          B=0    W              .
3070 175E2 D8           B=A    A              .
3071 175E4 866          ?ST=0  sODDST        Not odd start?
3072 175E7 01           GOYES  NIBL57         .
3073 175E9 E5           B=B+1  A             Increment working nib count
3074 175EB 879          ?ST=1  sREAD         Reading?  (then ignore extra nib)
3075 175EE 90           GOYES  NIBL57         .
3076 175F0 14A          A=DAT0 B             Copy odd nib from I/O buff to dest
3077 175F3 1590         DAT1=A 1              .
3078 175F7 847   NIBL57 ST=0   sODDEN        Clear "Odd end" flag
3079 175FA 822          SB=0                 Compute whole working bytes
3080 175FD 81D          BSRB                  .
3081 17600 832          ?SB=0                Not odd end?
3082 17603 50           GOYES  NIBL60         .
3083 17605 857          ST=1   sODDEN        Set "Odd end" flag
3084              *
3085              **    Save status from HPIL routines
3086              **    Read/write working number of whole bytes
3087              *
3088              *NIBL60 GOSUB  DRO1EX        Save D0, D1
3089              *       D1=(5) /NSTAT        Save status from HPIL pot shots
3090              *       GOSUB  STATSV         .
3091              *       GOSUB  DRO1EX        Restore D0, D1
3092
3093 17608 879   NIBL60 ?ST=1  sREAD          Reading?
3094 1760B F1           GOYES  NIBL67
3095 1760D 580          GONC   NIBL65         BET
3096              *
3097 17610 8E00  NIBL64 GOSUBL =WRBYTD        WRITE A BYTE
          00
3098 17616 CD    NIBL65 B=B-1  A              Subtract byte
3099 17618 57F          GONC   NIBL64         Loop if more
3100 1761B 431          GOC    NIBL68         BET (Last whole byte written)
3101              *
3102 1761E 8E00  NIBL66 GOSUBL =RDBYTA        READ A BYTE
```

```
            00
3103 17624 149          DAT1=A B              Write to buffer
3104 17627 171          D1=D1+ 2               .
3105 1762A CD    NIBL67 B=B-1  A              Subtract byte
3106 1762C 51F          GONC   NIBL66         Loop if more
3107               *
3108               **     Restore status
3109               **     Read/write odd nib if needed
3110               *
3111               *NIBL68 GOSUB  DR01EX        Save D0, D1
3112               *       D1=(5) /NSTAT        Restore status
3113               *       GOSUB  STATRS         .
3114               *       GOSUB  DR01EX        Restore D0, D1
3115
3116 1762F 867    NIBL68 ?ST=0  sODDEN         Not odd end?
3117 17632 81           GOYES  NIBL70
3118 17634 869          ?ST=0  sREAD          Writing?
3119 17637 C0           GOYES  NIBL69          .
3120 17639 14A          A=DAT0 B              Read final nib from I/O buff
3121 1763C 1590         DAT1=A 1              Write final nib to output buffer
3122 17640 590          GONC   NIBL70         BET
3123 17643 14B   NIBL69 A=DAT1 B              Read final nibble from output buff
3124 17646 1580         DAT0=A 1              Write final nib to I/O buffer
3125 1764A 84B   NIBL70 ST=0   11             Set up for UPCPOS
3126               *
3127               *  NOTE: UPCPOS requires R1 = rec len for fixed length data file
3128               *    and assumes S10, S11
3129               *
3130 1764D 8E00         GOSUBL =UPCPOS        Finish updating current pos
            00
3131 17653 867          ?ST=0  sODDEN         Not odd end?
3132 17656 A0           GOYES  NIBL90
3133 17658 D0           A=0    A              Increment current position 1 nib
3134 1765A E4           A=A+1  A               .
3135 1765C 798E         GOSUB  FIBUPD          . note: modifies S7
3136 17660 866   NIBL90 ?ST=0  sODDST         Not "Odd Start"?
3137 17663 01           GOYES  NIBL95         Return
3138               *
3139               ** Remove extra nibble from start of buffer
3140               *
3141 17665 733C         GOSUB  D0=/BX         A = Buffer start + 1
3142 17669 D6           C=A    A               .
3143 1766B E4           A=A+1  A               .
3144 1766D E2           C=C-A  A              C = -1
3145 1766F 7410         GOSUB  OBEDIT         Remove nibble
3146 17673 6CFE  NIBL95 GOTO   RSTKR+         Return (no error possible)
3147
3148               *
3149 17677 0            CON(1) =FIXSPC
3150 17678              BSS    16-1
3151               *
```

```
3152                OBEDIT  STITLE Edit Output Buffer
3153                ********************************************************************
3154                ********************************************************************
3155                **
3156                ** Name:(S) OBEDIT  -  Edit Output Buffer
3157                **
3158                ** Category:   FILUTL
3159                **
3160                ** Purpose:
3161                **      Move the trailing portion of the output buffer, between
3162                **      a specified address and (AVMEMS), up or down by a given
3163                **      offset.  Update AVMEMS and perform memory check when
3164                **      offset is positive.
3165                **
3166                ** Entry:
3167                **      A(A)    =  Start of block to move (SOURCE).
3168                **      C(A)    =  Offset of move (DEST - SOURCE).  If positive,
3169                **                 memory check will be performed.
3170                **      P       =  0 if leeway is desired should a memory check
3171                **                   be performed.
3172                ** Exit:
3173                **      P       =  0
3174                **     Carry clear:
3175                **      A(A)    =  Start of block to move (SOURCE).
3176                **      B(A)    =  Length of block moved (old (AVMEMS)-SOURCE).
3177                **      C(A)    =  DESTination of move (new start of block).
3178                **      (AVMEMS) updated, now old (AVMEMS) + offset.
3179                **     Carry set:
3180                **      C(3-0) =  eMEM error code (Insufficient memory)
3181                **
3182                ** Calls:      MEMCL+, MOVE*M
3183                **
3184                ** Uses.......
3185                **  Exclusive: A(A), B(A), C(A),     D1, P
3186                **  Inclusive: A(A), B(A), C(A), D0, D1, P
3187                **
3188                ** Stk lvls:   1
3189                **
3190                ** History:
3191                **
3192                **    Date     Programmer             Modification
3193                **  --------   ----------   ----------------------------------
3194                **  09/21/82      FH        Designed and coded.
3195                **
3196                ********************************************************************
3197                ********************************************************************
3198 17687 D5    =OBEDIT B=C     A               B = offset
3199 17689 C2            C=A+C   A               C = destination
3200 1768B 8BE           ?C<=A   A               Contraction?
3201 1768E 21            GOYES   OBED20
3202 17690 130           D0=A                    Save source
3203 17693 8F00          GOSBVL  =MEMCL+         Check memory
           000
3204 1769A 400           RTNC
3205 1769D 132           ADOEX                   Recall source
```

```
3206 176A0 1F49  OBED20 D1=(5) =AVMEMS       Update AVMEMS
           5F2
3207 176A7 147           C=DAT1 A             .
3208 176AA C9            C=C+B  A             .  (new av mem start)
3209 176AC 145           DAT1=C A             .
3210 176AF DD            CBEX   A            B = LEN
3211 176B1 E1            B=B-C  A             .  (old av mem start)
3212 176B3 E8            B=B-A  A             .  (old av mem start) - (source)
3213 176B5 C2            C=C+A  A            C = DEST = offset + SOURCE
3214 176B7 8D00  move*m  GOVLNG =MOVE*M      Go do it
           000
```

```
3215            RECNIB  STITLE Recall Into A Reg the Nibs Read
3216            ********************************************************************
3217            ********************************************************************
3218            **
3219            ** Name:    RECNIB  -  Recall Into A Reg the Nibs Read
3220            ** Name:    RECAL-  -  Recall Into A Reg the Nibs Read
3221            **
3222            ** Category:   FILUTL
3223            **
3224            ** Purpose:
3225            **      Reads into the A register the last n nibs of the
3226            **      output buffer, where n = min(R3,16).  R3 contains
3227            **      the number of nibs READNB appended to the output
3228            **      buffer.  For RECAL-, the R3 value is in C(A).
3229            **
3230            ** Entry:
3231            **    RECNIB:
3232            **      R3(A)  =  Number of nibs READNB has appended to the
3233            **                  output buffer.
3234            **    RECAL-:
3235            **      C(A)   =  Number of nibs READNB has appended to the
3236            **                  output buffer.
3237            **
3238            ** Exit:
3239            **      A      =  Up to last 16 nibs of data read
3240            **      C(A)   =  Address of old av mem start
3241            **      D0     =  Address of old av mem start
3242            **      Carry  =  Clear.
3243            **
3244            ** Calls:      D1@AVS
3245            **
3246            ** Uses.......
3247            **  Inclusive: A,C(A),D0,D1
3248            **
3249            ** Stk lvls:   1
3250            **
3251            ********************************************************************
3252            ********************************************************************
3253 176BE 11B   RECNIB C=R3                    Recall #nibs
3254 176C1 8F00  RECAL- GOSBVL =D1@AVS          Set D1 = AVMEMS, A=(AVMEMS)
          000
3255 176C8 EE            C=A-C  A              Back up before av mem st
3256 176CA 134           D0=C                  D0 @ new av mem start
3257 176CD 1527          A=DAT0 W              Read data after new end of buffer
3258 176D1 03            RTNCC
```

```
3259            TXT->B  STITLE Transform TEXT to BASIC
3260            ******************************************************************
3261            ******************************************************************
3262            **
3263            ** Name:    TXT->B  -  Transform TEXT to BASIC
3264            **
3265            ** Category:   LOCAL
3266            **
3267            ** Purpose:
3268            **      Read line from source TEXT file, transform into BASIC,
3269            **      leave it in output buffer.  No messages are directly
3270            **      issued by this routine except parse warnings (Overflow
3271            **      or Underflow with IEEE math setting 2) which are
3272            **      beyond its control (issued by parse routines).
3273            **
3274            ** Entry:
3275            **      R4(15,14)     =  Source FIB#
3276            **      Input, output buffers collapsed to SYSEN
3277            **      At least 150 bytes + LEEWAY available memory guaranteed
3278            **      /LINE#        =  0 or previously returned BCD line #
3279            **      /SFIB#        =  Source FIB#
3280            **      /OPTN         =  Option from extended TRANSFORM statement
3281            **                       execution; 0 if from normal TRANSFORM
3282            **      /PARM1,/PARM2 = Destination file create parameters from
3283            **                       extended TRANSFORM statement execution;
3284            **                       0 if from normal TRANSFORM
3285            **      P             =  0
3286            **
3287            ** Exit:
3288            **      OUTBS  @  Start of transformed line.  If original line
3289            **                was copied into available memory start, OUTBS
3290            **                may point immediately after the original line.
3291            **                Must be collapsed to /SYSEN if fatal error.
3292            **      AVMEMS @  End of transformed line unless fatal error.
3293            **                Must be collapsed to /SYSEN if fatal error.
3294            **      S7     =  1 iff end of file found on source file (sEOF)
3295            **      /LNLEN =  Full length in nibs of input line.  Unneeded
3296            **                if fatal error.
3297            **      /LINE# =  BCD line number of current line.  Used in
3298            **                reporting error messages.  If sequential
3299            **                line number is to be used, set to 0.
3300            **      P      =  0
3301            **     Carry clear:  Successful transformation
3302            **     Carry set:    Error occurred
3303            **      C(3-0) =  Error code
3304            **      C(S)   =  0 if error was fatal (unrecoverable).
3305            **             #  0 if error was recoverable.
3306            **
3307            ** Uses.......
3308            **      All CPU registers, S0-S11, S13, Statement and Function
3309            **      scratch RAM, SNAPBF, RSTKBF, /LNLEN, /LINE#, /FLAG,
3310            **      INBS, OUTBS, AVMEMS
3311            **
3312            ** Stk lvls:   6
3313            **
```

```
3314                ** Algorithm:
3315                **      Read text line into input/output buffer
3316                **      Append endline
3317                **      Save input line length in /LNLEN
3318                **      Skip over input buffer
3319                **      Parse
3320                **      Fetch line# from parsed string, save in R1
3321                **      If no error, then
3322                **         Save line # in /LINE#
3323                **         Update AVMEMS from D0
3324                **         Collapse input buffer
3325                **         Recall line length from /LNLEN, store in R2
3326                **         If line was truncated, go to 6.0
3327                **         Compute length of output buffer into R3
3328                **         Return CC
3329                **      If fatal error (no line #) then
3330                **         Return SC
3331                ** 6.0  Save error code
3332                **      Save line # in /LINE#
3333                **      Collapse output buffer
3334                **      Make input buffer output buffer
3335                **      Blow open 14-nib hole at buffer start
3336                **      Move line# into first 4
3337                **      Compute line length, store in next 2
3338                **      Write \! ? \ tag into last 8
3339                **      Change Endline to EOL token
3340                **      Compute length of output buffer into R3
3341                **      Recall line length from /LNLEN, store in R2
3342                **      Recall error code
3343                **      Return SC
3344                **
3345                ** History:
3346                **
3347                **     Date     Programmer            Modification
3348                **   --------   ----------   ---------------------------------
3349                **  06/12/82      FH        Designed and coded.
3350                **
3351                ********************************************************************
3352                ********************************************************************
3353 176D3 E210          REL(5) TXTCHB         Finish-up routine (Chain file)
         0
3354 176D8 6A10          REL(5) B->TXT         Inverse routine
         0
3355                *
3356                *  Call RDTEXT to read line from source file
3357                *  Save away full input line len
3358                *
3359 176DD 78AD =TXT->B GOSUB  RDTEXT          Read input line
3360 176E1 7803         GOSUB  TFUSLL         Save line length, set C(S) = 0
3361 176E5 70C8         GOSUB  TFUCCK         Return if fatal error or EOF
3362 176E9 7C62         GOSUB  LNOVF?         Test for line too long
3363 176ED 570          GONC   TXT-20          . branch if NO
3364 176F0 C0           A=A+B  A               . if YES, reset AVMEMS
3365 176F2 141          DAT1=A A               .  to truncate line
3366 176F5 1FAF  TXT-20 D1=(5) /FLAG          Store "Truncated Line" flag
```

```
            8F2
3367 176FC 1554         DAT1=C S                .
3368               *
3369               *  Add Endline to line
3370               *  Skip over input buffer
3371               *
3372 17700 31D0         LCHEX  0D               Append EOL
3373 17704 DA           A=C    A                .
3374 17706 7D41         GOSUB  WTAVN2           .
3375 1770A 400          RTNC                    Return if fatal error
3376 1770D 143          A=DAT1 A                Skip over input buffer by setting
3377 17710 1C4          D1=D1- (AVMEMS)-(OUTBS) .
3378 17713 141          DAT1=A A                .  OUTBS to AVMEMS
3379               *
3380               *     Set up parameters for LINE PARSE call
3381               *     Call LNPEXT
3382               *     Clear EOF status
3383               *
3384 17716 D0           A=0    A                Zero out AUTINC
3385 17718 1FBC         D1=(5) =AUTINC          .
            6F2
3386 1771F 1593         DAT1=A 4                .
3387 17723 7B6B         GOSUB  R<rstk           Hide my return address from parse
3388
3389               ****************************************************************
3390               *
3391               *  Parse the line.     Interface is:
3392               *
3393               *  Entry:
3394               *       INBS   -> Start of input line
3395               *       OUTBS  -> Start of available memory, destination for
3396               *                    parsed line
3397               *
3398               *  Exit:
3399               *       OUTBS  -> Start of parsed line
3400               *       AVMEMS -> End of parsed line
3401               *       S5     =  0 if no line number
3402               *              =  1 if line number present
3403               *    Carry clear:
3404               *       Successful parse (missing line number not allowed)
3405               *    Carry set:
3406               *       Mem error in parse
3407               *       C(3-0) =  Error code
3408               *
3409               ****************************************************************
3410 17727 8F00         GOSBVL =LNPEXT          Parse line
            000
3411 1772E 7999         GOSUB  CARYSV           Save carry
3412 17732 8AE          ?C#0   A                Error code supplied?
3413 17735 60           GOYES  TXT-40
3414 17737 3100         LC(2)  =eSYNTX          Supply default
3415 1773B 10A   TXT-40 R2=C                    Save carry, error code in R2
3416 1773E 847          ST=0   sEOF             Clear EOF status
3417 17741 754B         GOSUB  Rstk<r           Recall return address
3418 17745 11A          C=R2                    Line number missing?
```

```
3419 17748 AC2          C=0    S              .
3420 1774B 865          ?ST=0  5              .
3421 1774E 00           RTNYES                .
3422 17750 7182         GOSUB  TFUUD+         Update line number
3423 17754 101          R1=A                   . and save in R1
3424 17757 1FAF         D1=(5) /FLAG          Check "line too long" flag
           8F2
3425 1775E 14B          A=DAT1 B              .
3426 17761 3300         LC(4)  =eL2LNG        .
           00
3427 17767 90C          ?A#0   P              .
3428 1776A B0           GOYES  TXT-60          . and branch if set
3429 1776C 11A          C=R2                  Return if no error
3430 1776F A4E          C=C-1  S              .
3431 17772 500          RTNNC                 .
3432             *
3433             *  Collapse output buffer
3434             *  Add EOL token
3435             *  Reinstate input buffer as output buffer
3436             *
3437 17775 10A   TXT-60 R2=C                  Save updated error code
3438 17778 8F00         GOSBVL =OBCOLL        Collapse output buffer
           000
3439 1777F 1B6C         D0=(5) =INBS          Set (OUTBS) to (INBS)
           6F2
3440 17786 142          A=DAT0 A                      .
3441 17789 1C9          D1=D1- 5+(AVMEMS)-(OUTBS)     .
3442 1778C 141          DAT1=A A                      .
3443             *
3444             *  Remove ASCII line # and any trailing blanks
3445             *  Blow open hole for tokenized line #, length, and tag
3446             *  Write line #
3447             *  Compute length, save in R3, write to output buffer
3448             *  Write tag
3449             *  Recall input length, write to R2
3450             *
3451 1778F 131          D1=A                  D1 = start of line
3452 17792 7C72         GOSUB  SKIPLN         Skip line number
3453 17796 14B          A=DAT1 B              Skip following blank if any
3454 17799 3102         LCASC  \ \             .
3455 1779D 966          ?A#C   B               .  Not blank?
3456 177A0 50           GOYES  TXT-80          .
3457 177A2 171          D1=D1+ 2               .
3458 177A5 76BA  TXT-80 GOSUB  d0=obs         D0 = DEST   = line start + 12
3459 177A9 16B          D0=D0+ 12              .
3460 177AC 133          AD1EX                 A = SOURCE of move = after line#
3461 177AF 136          CD0EX                 C = OFFSET = DEST - SOURCE
3462 177B2 E2           C=C-A  A              .
3463 177B4 7FCE         GOSUB  OBEDIT         Remove ASCII line #
3464 177B8 AC2          C=0    S              .
3465 177BB 400          RTNC                  .
3466 177BE 7D9A         GOSUB  d0=obs         D0 at line start
3467 177C2 111          A=R1                  Recall and write line #
3468 177C5 1583         DAT0=A 4              .
3469 177C9 799A         GOSUB  oblcmp         Compute line length
```

```
3470 177CD D2          C=0    A              Subtract 4 for line#
3471 177CF 304         LCHEX  4               .
3472 177D2 EA          A=A-C  A               .
3473 177D4 163         DO=DO+ 4               Write length offset
3474 177D7 148         DATO=A B               .
3475 177DA 3500        LC(6)  \ ?\*256+(=t!) Write tag \ ?<t!>\
           0000
3476 177E2 161         DO=DO+ 2               .
3477 177E5 15C5        DATO=C 6               .
3478 177E9 3100        LC(2)  =tEOL           Add EOL token
3479 177ED DA          A=C    A               .
3480 177EF 7460        GOSUB  WTAVN2          .
3481 177F3 400         RTNC                   . and return if fatal error
3482              *
3483              *  Return error condition
3484              *
3485 177F6 11A         C=R2                   Recall error code
3486 177F9 AC2         C=0    S               Set nonfatal status
3487 177FC B46         C=C+1  S               .
3488 177FF 02          RTNSC                  Return
3489
3490              ****************************************************************
3491              ****************************************************************
3492              **
3493              ** Name:    TXTCHB  -  Chain BASIC Dest File After TRANSFORM
3494              **
3495              ** Category:   LOCAL
3496              **
3497              ** Purpose:
3498              **      To finish up the destination BASIC file after all TEXT
3499              **      lines have been transformed into BASIC.  There are
3500              **      several cases to be dealt with:
3501              **
3502              **     End of a Dry Run (always out-of-place transform):
3503              **      If the destination file is on an external medium, a
3504              **      first pass or "dry run" is conducted without creating
3505              **      the dest file, in order to determine its necessary data
3506              **      size.  This routine calculates the needed parameters
3507              **      to create the file (see CRTF utility), and stores them
3508              **      in /PARM1 and /PARM2.
3509              **
3510              **     End of a Normal Run, Out-of-Place Transform:
3511              **      If the destination file type requires a subheader or
3512              **      Implementation field, it must be properly initialized
3513              **      since CRTF stored a default value in it when the file
3514              **      was created (see CRTF utility).  For example, if the
3515              **      destination is a BASIC file, hex value 0000000000F
3516              **      must be written to the header to indicate that the link
3517              **      chain heads have not been computed for this file (file
3518              **      has not been "chained").  If the file is in memory, the
3519              **      the proper link chain heads are computed and written to
3520              **      the subheader.  File data, such as links between sub-
3521              **      programs in BASIC files, may need to be updated.
3522              **
3523              **     End of a Normal Run, In-Place Transform:
```

```
3524            **      If the source file type had a subheader or Implementa-
3525            **      tion field, it must be removed.  If the destination
3526            **      file type requires a subheader or Implementation field,
3527            **      it must be inserted after the file header and set to
3528            **      the proper value.  File data, such as links between
3529            **      subprograms in BASIC files, may need to be updated.
3530            **
3531            **    End of an Inverse Transformation (Always In-Place):
3532            **      If the source file type had a subheader or Implementa-
3533            **      tion field, it it is still there but may need to be
3534            **      updated to reflect the new state.  File data, such as
3535            **      links between subprograms in BASIC files, may need
3536            **      to be updated.
3537            **
3538            ** Entry:
3539            **      Output buffer collapsed (OUTBS, AVMEMS point to SYSEN)
3540            **      R4(15-14) =  FIB# of destination file. Each line of the
3541            **                   source file (including EOF) has been read,
3542            **                   transformed, and written to the dest file.
3543            **                   The End of Data field in the FIB is set to
3544            **                   this new end of file and, if the dest file
3545            **                   is in memory, any excess nibs beyond the
3546            **                   end of file have been removed from the
3547            **                   file chain.  File is now rewound.
3548            **      S5      =  1 iff transformation is in place (sTFINP)
3549            **      S6      =  1 iff at end of inverse transformation
3550            **                   (sTFINV)
3551            **      S9      =  1 iff at end of dry run (sDRYRN)
3552            **      P       =  0
3553            **
3554            ** Exit:
3555            **      P       =  0
3556            **     Carry clear:
3557            **      No error
3558            **     Carry set:
3559            **      C(3-0) =  Error code (will be treated as fatal error,
3560            **                with no possibility of recovering dest file)
3561            **
3562            ** Calls:       WTAVWP, RPLSBH, LOCFHD, CHAIN-
3563            **
3564            ** Uses.......
3565            **  Inclusive: May use any CPU register, S10
3566            **
3567            ** Stk lvls:   6 (max)
3568            **
3569            ** Algorithm:
3570            **      If Dry Run, then
3571            **        Read file size in /DLEN
3572            **        Store file size in /PARM1
3573            **      Else
3574            **        If Inverse Transform or Out-of-place Transform, then
3575            **          Rewrite "Unchained" value to existing subheader
3576            **        Else
3577            **          Insert new subheader with "Unchained" value
3578            **        Call CHAIN- to chain file links
```

```
3579                **
3580                ************************************************************************
3581                ************************************************************************
3582                *
3583                *  Insert new subheader
3584                *
3585 17801 869    TXTCHB ?ST#1  sDRYRN         NOT Dry Run?
3586 17804 81             GOYES  TXTC20
3587                *
3588 17806 1FAD   SETLEN D1=(5) /DLEN          Read length
           8F2
3589 1780D AF0            A=0    W              .
3590 17810 143            A=DAT1 A              .
3591 17813 1DBE           D1=(2) /PARM1         Store into /PARM1
3592 17817 141            DAT1=A A               .
3593 1781A 03             RTNCC
3594                *
3595 1781C 2B     TXTC20 P=     12-1            Set A(11-0) = #F00000000000
3596 1781E AF0            A=0    W              .
3597 17821 A0C            A=A-1  P              .
3598 17824 D2             C=0    A              Set up insert unless inverse tfm
3599 17826 866            ?ST=0  sTFINV          . NOT inverse Xform completed?
3600 17829 50             GOYES  TXTC40          .
3601 1782B 809            C+P+1                  .
3602 1782E 10B    TXTC40 R3=C                   .
3603 17831 7420           GOSUB  WTAVWP          Write to ouput buffer
3604 17835 400            RTNC                    .
3605                *
3606                *  NOTE: For out-of-place memory Xform, RPLSBH always REPLACES
3607                *        the old subheader with the new one
3608                *
3609 17838 7F51           GOSUB   RPLSBH         Insert/replace subheader
3610 1783C 400            RTNC
3611 1783F 7759           GOSUB  LOCFHD          Locate file header
3612 17843 87A            ?ST=1  =sI/OBF         External file?
3613 17846 90             GOYES  TXTCcc
3614 17848 8F00           GOSBVL =CHAIN-         Chain file
           000
3615 1784F 03     TXTCcc RTNCC
```

```
3616            WTAVWP  STITLE Write A(WP) to Avmem Start
3617            ********************************************************************
3618            *
3619            *  WTAVWP  -  Write A(WP) to Avmem Start, Update AVMEMS
3620            *
3621            *  Entry:
3622            *       P      =  # nibs to write - 1
3623            *       A(WP)  =  Data to write
3624            *
3625            *  Exit:
3626            *       P      =  0
3627            *       A      =  Data to write
3628            *       D0     @  Old av mem start
3629            *       D1     =  AVMEMS
3630            *       C(S)   =  0
3631            *
3632            *  Uses:  P,D1,D0,C(A),A,B(A),R0
3633            *
3634 17851 D0    WTAVFF A=0    A              Set up write of FFFF
3635 17853 CC           A=A-1  A               .
3636 17855 22    WTAVN4 P=     2              Set P=4-1
3637 17857 0C    WTAVN2 P=P+1                  . Set P=2-1
3638 17859 100   WTAVWP R0=A                  Save data
3639 1785C D2           C=0    A              C  =  #nibs
3640 1785E 809          C+P+1                  .
3641 17861 7791         GOSUB  OBAPND         Blow open space at end of buffer
3642 17865 AC2          C=0    S              Set C(S) to zero for return
3643 17868 400          RTNC                  Return if memerr
3644 1786B 110          A=R0                  Recall data
3645 1786E 1507         DAT0=A W              Write data
3646 17872 03    WTAVcc RTNCC
```

```
3647            B->TXT  STITLE Transform BASIC to TEXT
3648            ******************************************************************
3649            ******************************************************************
3650            **
3651            ** Name:    B->TXT  -  Transform BASIC to TEXT
3652            **
3653            ** Category:   LOCAL
3654            **
3655            ** Purpose:
3656            **      Read line from source BASIC file, transform into TEXT,
3657            **      leave it in output buffer.  No messages are directly
3658            **      issued by this routine.
3659            **
3660            ** Entry:
3661            **      R4(15,14)     =  Source FIB#
3662            **      Input, output buffers collapsed to SYSEN
3663            **      At least 150 bytes + LEEWAY available memory guaranteed
3664            **      /LINE#        =  0 or previously returned BCD line #
3665            **      /SFIB#        =  Source FIB#
3666            **      /OPTN         =  Ignored
3667            **      /PARM1,/PARM2 = Ignored
3668            **      P             =  0
3669            **
3670            ** Exit:
3671            **      OUTBS  @  Start of transformed line.  If original line
3672            **                was copied into available memory start, OUTBS
3673            **                may point immediately after the original line.
3674            **                Must be collapsed to /SYSEN if fatal error.
3675            **      AVMEMS @  End of transformed line unless fatal error.
3676            **                Must be collapsed to /SYSEN if fatal error.
3677            **      S7     =  1 iff end of file found on source file (sEOF)
3678            **      /LNLEN =  Full length in nibs of input line.  Unneeded
3679            **                if fatal error.
3680            **      /LINE# =  BCD line number of current line.  Used in
3681            **                reporting error messages.  If sequential
3682            **                line number is to be used, set to 0.
3683            **      P      =  0
3684            **     Carry clear:  Successful transformation
3685            **     Carry set:    Error occurred
3686            **      C(3-0) =  Error code
3687            **      C(S)   =  0 if error was fatal (unrecoverable).
3688            **             #  0 if error was recoverable.
3689            **
3690            ** Calls:      LDSST1, STATSV, STATRS, TFURSF
3691            **
3692            ** Uses:
3693            **      All CPU registers, S0-S11, S13, Statement and Function
3694            **      scratch RAM, SNAPBF, RSTKBF, /LNLEN, /LINE#, /FLAG,
3695            **      INBS, OUTBS, AVMEMS
3696            **
3697            ** Stk lvls:   6
3698            **
3699            ** Algorithm:
3700            **      Space OUTBS out 4 nibs for line length header
3701            **      Save max input buf len in R1: min(256 bytes, av mem)
```

```
3702                ** Save status
3703                ** Locate file
3704                ** If external file, then
3705                **    Error exit: not implemented
3706                ** Calculate current position of file somehow
3707                ** If EOF then
3708                **    Set EOF flag
3709                **    Inp buff len = 0
3710                **    Output buff len = 0
3711                **    Return carry clear
3712                ** Set INBS to current position
3713                ** Calculate and save input line len
3714                ** Set up parameters for Decompile
3715                ** Call Decompile
3716                ** Recall input len, move to R2
3717                ** Save len of line read in R3
3718                ** Compare line begin with [! ? ] tag
3719                ** If equal, then
3720                **    Move OUTBS out 8 nibs
3721                **    Reduce count in R3 by 8
3722                ** Recall line len in nibs
3723                ** Convert to bytes
3724                ** Store at start of OUTBS
3725                ** Add 4 nibs to total for output buffer len
3726                ** Store in R3
3727                ** Return with no error
3728                **
3729                ** History:
3730                **
3731                **     Date     Programmer            Modification
3732                **   --------   ----------   ---------------------------------
3733                **  06/14/82      FH        Designed and coded.
3734                **
3735                ********************************************************************
3736                ********************************************************************
3737 17874 4010         REL(5) B->CHT          Finish-up routine
         0
3738 17879 46EF         REL(5) TXT->B          Inverse routine
         F
3739                *
3740                *  Read input line
3741                *  Save input line length
3742                *  If EOF then write FFFF to output buffer, return
3743                *
3744 1787E 7D7B =B->TXT GOSUB  RDBAS           Read input line
3745 17882 7761         GOSUB  TFUSLL          Save line length, set C(S)=0
3746 17886 400          RTNC                   Return if fatal error
3747 17889 877          ?ST=1  =sEOF           EOF?
3748 1788C 5C           GOYES  WTAVFF          If so, write EOF
3749                *
3750                *  Space OUTBS, AVMEMS out 4 beyond current av mem start
3751                *    (possible without memcheck since RDBAS chacks leeway)
3752                *  Set INBS to current position
3753                *
3754                *  Set up parameters for Decompile
```

```
3755            *  Call Decompile
3756            *  Clear EOF flag
3757            *
3758 1788E 7BD9        GOSUB  obprd         Space output buffer beyond line hdr
3759 17892 D2          C=0    A               .
3760 17894 304         LC(1)  4               .
3761 17897 C2          C=C+A  A               .
3762 17899 145         DAT1=C A               . (update AVMEMS)
3763 1789C 1C4         D1=D1- (AVMEMS)-(OUTBS) .
3764 1789F 145         DAT1=C A               . (update OUTBS)
3765 178A2 11B         C=R3                  Set D1, D0 to start of line
3766 178A5 135         D1=C                   . and set D1 to input pointer
3767 178A8 134         D0=C                   .
3768 178AB 7C21        GOSUB  TFUUDL         Update line number
3769
3770            ********************************************************************
3771            *
3772            *       Decompile Line  --  Interface is:
3773            *
3774            *  Entry:
3775            *       D1      -> Start of input line
3776            *
3777            *  Exit:
3778            *       OUTBS   -> Start of decompiled line
3779            *       AVMEMS  -> End of decompiled line
3780            *    Carry clear:
3781            *       Successful decompile (missing line number allowed)
3782            *    Carry set:
3783            *       Mem error in decompile
3784            *       C(3-0) =  Error code
3785            *       S5     =  0
3786            *
3787            ********************************************************************
3788 178AF 8F00          GOSBVL =LDCEXT        Decompile line
            000
3789 178B6 847         ST=0   sEOF           Clear EOF flag
3790 178B9 AC2         C=0    S              Flag fatal error
3791 178BC 400         RTNC
3792            *
3793            *  Update AVMEMS, check for "! ? " tag
3794            *
3795            *
3796 178BF 7AA9        GOSUB  obprd          Update AVMEMS to D0
3797 178C3 132         ADOEX                  .
3798 178C6 141         DAT1=A A               .
3799 178C9 135         D1=C                  D1 @ start of line in output buff
3800 178CC 7241        GOSUB  SKIPLN         Skip line number
3801 178D0 100         R0=A                   . and save in R0
3802 178D3 AF0         A=0    W              Read next 5 chars
3803 178D6 15B9        A=DAT1 5*2             .
3804 178DA AF2         C=0    W              Load tag ( set C(S)=0 for below )
3805 178DD 3902        LCASC  \ ? ! \         .
            1202
            F302
3806 178E9 976         ?A#C   W              NOT equal?
```

```
3807 178EC 50          GOYES  B->T70
3808 178EE 177         D1=D1+ 4*2            Space beyond \? ! \ to remove it
3809            *
3810            *  Set SOURCE to blank after line #
3811            *  Set DEST to 5th character position of line
3812            *  Compute OFFSET
3813            *  Edit output buffer
3814            *  Read input line #
3815            *  Convert to 4 decimal ascii chars w/ leading zeroes
3816            *  Write to start of output buffer
3817            *
3818 178F1 133   B->T70 AD1EX                 A = SOURCE
3819 178F4 7769        GOSUB  dO=obs         C = OFFSET = (OUTBS) + 8 - SOURCE
3820 178F8 E2          C=C-A  A               .
3821 178FA 137         CD1EX                  .
3822 178FD 177         D1=D1+ 8               .
3823 17900 137         CD1EX                  .
3824 17903 708D        GOSUB  OBEDIT         Edit output buffer
3825 17907 400         RTNC                  Return fatal error if carry set
3826 1790A 23          P=     4-1            Set digit count for conversion
3827 1790C 80FF        CPEX   15              . and restore P=0
3828 17910 110         A=RO                  Recall line number
3829 17913 7138        GOSUB  HEXASC         Convert with leading zeroes
3830 17917 7449        GOSUB  dO=obs         Write to output buffer
3831 1791B 1587        DAT0=A 8               .
3832            *
3833            *  Compute output line length in bytes
3834            *  Store at start of line, update OUTBS
3835            *
3836 1791F AF0         A=0    W              A = line data length in bytes
3837 17922 7049        GOSUB  oblcmp          .
3838 17926 81C         ASRB                  .
3839 17929 7848        GOSUB  LIF>NB         Test if we have an odd # of bytes
3840 1792D 511         GONC   B->T80          . and jump if NOT odd
3841 17930 101         R1=A                  Save away #bytes in R1
3842 17933 D0          A=0    A              Write 00 at end of avail memory
3843 17935 7E1F        GOSUB  WTAVN2          .
3844 17939 400         RTNC                  . and error out if mem err
3845 1793C 111         A=R1                  Recall # bytes
3846 1793F 71E0  B->T80 GOSUB  SWPBYT        Reverse bytes for line header
3847 17943 1C4         D1=D1- (AVMEMS)-(OUTBS)  Set OUTBS back to true start
3848 17946 147         C=DAT1 A                .
3849 17949 137         CD1EX                   .
3850 1794C 1C3         D1=D1- 4                .
3851 1794F 1593        DAT1=A 4                . write line header
3852 17953 137         CD1EX                   .
3853 17956 145         DAT1=C A                . update OUTBS
3854
3855            ******************************************
3856            *                                        *
3857            *   NOTE:  FALLS INTO FOLLOWING CODE     *
3858            *                                        *
3859            ******************************************
3860
3861            ************************************************************
```

```
3862                **
3863                ** Name:    LNOVF?  -  Truncate ASCII Line
3864                **
3865                ** Purpose:
3866                **      Test if line in output buffer is longer than 120 chars.
3867                **
3868                ** Entry:
3869                **      P       =  0
3870                **
3871                ** Exit:
3872                **      P       =  0
3873                **      A(A)    =  Avail mem start
3874                **      B(A)    =  Char limit in nibs ( = 120*2)
3875                **     Carry clear:
3876                **      C(S)    =  0
3877                **     Carry set:
3878                **      C(S)    =  1
3879                **      C(3-0)  =  Error code: eL2LNG
3880                **
3881                ** Calls:       oblcmp
3882                **
3883                ** Uses.......
3884                **  Inclusive: A(A), B(A), C, D1
3885                **
3886                ** Stk lvls:   2
3887                **
3888                ********************************************************************
3889
3890                ******************************************
3891                *                                        *
3892                * NOTE:  THIS CODE FALLEN INTO FROM ABOVE *
3893                *                                        *
3894                ******************************************
3895
3896 17959 AF2    LNOVF? C=0     W               Load limit
3897 1795C 310F        LC(2)  120*2             .
3898 17960 D5          B=C    A                 Compute line length
3899 17962 7009        GOSUB  oblcmp            .
3900 17966 DE          ACEX   A                 .
3901 17968 8B9         ?C<=B  A                 Not too long?
3902 1796B 86          GOYES  RPLScc            . rtncc
3903 1796D B46         C=C+1  S                 Set flag
3904 17970 3300        LC(4)  =eL2LNG
           00
3905 17976 02          RTNSC
3906
3907                ********************************************************************
3908                ********************************************************************
3909                **
3910                ** Name:    B->CHT  -  Finish TEXT File After Xform from BASIC
3911                **
3912                ** Category:   LOCAL
3913                **
3914                ** Purpose:
3915                **      To finish up the destination TEXT file after all BASIC
```

```
3916            **      lines have been transformed into TEXT.  There are
3917            **      several cases to be dealt with:
3918            **
3919            **    End of a Dry Run (always out-of-place transform):
3920            **      If the destination file is on an external medium, a
3921            **      first pass or "dry run" is conducted without creating
3922            **      the dest file, in order to determine its necessary data
3923            **      size.  This routine calculates the needed parameters
3924            **      to create the file (see CRTF utility), and stores them
3925            **      in /PARM1 and /PARM2.
3926            **
3927            **    End of a Normal Run, Out-of-Place Transform:
3928            **      If the destination file type requires a subheader or
3929            **      Implementation field, it must be properly initialized
3930            **      since CRTF stored a default value in it when the file
3931            **      was created (see CRTF utility).  Since a TEXT file
3932            **      has no subheader or implementation field (in memory),
3933            **      this case is trivial.
3934            **
3935            **    End of a Normal Run, In-Place Transform:
3936            **      If the source file type had a subheader or Implementa-
3937            **      tion field, it must be removed.  If the destination
3938            **      file type requires a subheader or Implementation field,
3939            **      it must be inserted after the file header and set to
3940            **      the proper value.  File data, such as links between
3941            **      subprograms in BASIC files, may need to be updated.
3942            **      In the case of BASIC to TEXT, the source file's sub-
3943            **      header must be removed.
3944            **
3945            **    End of an Inverse Transformation (Always In-Place):
3946            **      If the source file type had a subheader or Implementa-
3947            **      tion field, it it is still there but may need to be
3948            **      updated to reflect the new state.  File data, such as
3949            **      links between subprograms in BASIC files, may need
3950            **      to be updated (this is true for BASIC to TEXT inverted
3951            **      back to BASIC).
3952            **
3953            ** Entry:
3954            **      Output buffer collapsed (OUTBS, AVMEMS point to SYSEN)
3955            **      R4(15-14) =  FIB# of destination file. Each line of the
3956            **                   source file (including EOF) has been read,
3957            **                   transformed, and written to the dest file.
3958            **                   The End of Data field in the FIB is set to
3959            **                   this new end of file and, if the dest file
3960            **                   is in memory, any excess nibs beyond the
3961            **                   end of file have been removed from the
3962            **                   file chain.  File is now rewound.
3963            **      S5      =  1 iff transformation is in place (sTFINP)
3964            **      S6      =  1 iff at end of inverse transformation
3965            **                   (sTFINV)
3966            **      S9      =  1 iff at end of dry run (sDRYRN)
3967            **      P       =  0
3968            **
3969            ** Exit:
3970            **      P       =  0
```

```
3971                **      Carry clear:
3972                **       No error
3973                **      Carry set:
3974                **       C(3-0) =  Error code (will be treated as fatal error,
3975                **                 with no possibility of recovering dest file)
3976                **
3977                ** Calls:      RPLSBH, SETLEN
3978                **
3979                ** Uses.......
3980                **  Inclusive: May use any CPU register, S10
3981                **
3982                ** Stk lvls:   6 (max)
3983                **
3984                ** Algorithm:
3985                **       If Dry Run, then
3986                **         Read file size in /DLEN
3987                **         Convert size to bytes
3988                **         Store file size in /PARM1
3989                **       Else
3990                **         If Transform Out-of-place or Inverse Transform, then
3991                **           Return OK
3992                **         Else {In-place Transform}
3993                **           Delete old BASIC subheader
3994                **
3995                ********************************************************************
3996                ********************************************************************
3997
3998                *
3999 17978 869    B->CHT ?ST#1  sDRYRN          NOT Dry Run?
4000 1797B E0           GOYES  B->C20
4001 1797D 758E         GOSUB  SETLEN         Set file length into A
4002 17981 81C          ASRB                  Convert len to bytes
4003 17984 141          DAT1=A A              Store in /PARM1 for create
4004 17987 03           RTNCC
4005                *
4006 17989 865    B->C20 ?ST=0  sTFINP          Transform NOT in place?
4007 1798C 74           GOYES  RPLScc         RTNCC
4008 1798E 876          ?ST=1  sTFINV         Inverse tansform?
4009 17991 24           GOYES  RPLScc         RTNCC
4010 17993 D2           C=0    A              Delete subheader
4011 17995 30C          LC(1)  (oFLSTr)-(oSUBLn)  .
4012 17998 10B          R3=C
4013                *       ************************************
4014                *       *                                  *
4015                *       *  NOTE:  FALL INTO FOLLOWING CODE  *
4016                *       *                                  *
4017                *       ************************************
```

```
4018            RPLSBH  STITLE Replace Subheader
4019
4020
4021            *       ******************************************
4022            *       *                                        *
4023            *       *  NOTE:  THIS IS FALLEN INTO FROM ABOVE  *
4024            *       *                                        *
4025            *       ******************************************
4026
4027
4028            ********************************************************************
4029            ********************************************************************
4030            **
4031            ** Name:(S) RPLSBH  -  Replace Memory File Subheader
4032            **
4033            ** Category:   FILUTL
4034            **
4035            ** Purpose:
4036            **      Replaces the subheader of a memory file with the data
4037            **      stored in the output buffer.  For external files, write
4038            **      the output buffer data to the subheader area of the
4039            **      file.  Does NOT update the subheader length field of
4040            **      the FIB, but for memory files it updates the Data Begin
4041            **      field.  If out-of-place transform in memory file, it
4042            **      replaces the old subheaderunconditionally with the new
4043            **      subheader in output buffer.
4044            **
4045            ** Entry:
4046            **      R4(15-14) = FIB# of dest file; file rewound.
4047            **      R3(A)  =  Length of old subheader
4048            **      P      =  0
4049            **      S5     =  1 iff In-place Transform (sTFINP)
4050            **      Output buffer contains new subheader
4051            **
4052            ** Exit:
4053            **      P      =  0
4054            **      R4(15-14) =  File FIB#
4055            **     Carry set:
4056            **      C(3-0) =  Error code; insufficient memory
4057            **
4058            ** Calls:      LOCFIL, RPLLI*, FIB#RS
4059            **
4060            ** Uses.......
4061            **  Inclusive: A,B,C,D(S),D(7-0),R0,R1,R2,R3,D0,D1
4062            **
4063            ** Stk lvls:   4
4064            **
4065            ** NOTE:
4066            **      File is ASSUMED to reside in memory (internal file).
4067            **
4068            **
4069            ** Algorithm:
4070            **      Adjust FIB pointers to make old subheader appear to be
4071            **           first line
4072            **      Replace this line with new subheader
```

```
4073            **      Adjust FIB pointers beyond new subheader again
4074            **
4075            ** History:
4076            **
4077            **    Date     Programmer            Modification
4078            **  --------   ----------   ----------------------------------
4079            **  10/04/82      FH        Designed and coded
4080            **
4081            ********************************************************************
4082            ********************************************************************
4083 1799B 7A78 =RPLSBH GOSUB  LOCFI#          Locate file
4084 1799F D2           C=0    A               Set external file to subheader
4085 179A1 145          DAT1=C A                .
4086 179A4 875          ?ST=1  sTFINP          Transformation in place?
4087 179A7 E0           GOYES  RPLS20           .
4088 179A9 87A          ?ST=1  =sI/OBF         External file?
4089 179AC 90           GOYES  RPLS20
4090 179AE 74B8         GOSUB  oblcmp          Compute new subheader length
4091 179B2 103          R3=A                    . and use that as prev len
4092 179B5 727B  RPLS20 GOSUB  WRITNB          Write/replace subheader
4093 179B9 400          RTNC                    .
4094 179BC 7958         GOSUB  LOCFI#          Locate file
4095 179C0 87A          ?ST=1  =sI/OBF         Return OK if external
4096 179C3 01           GOYES  RPLScc
4097 179C5 11B          C=R3                   Recall offset
4098 179C8 CA           A=A+C  A               Update data start
4099 179CA 1CF          D1=D1- 16                       .
4100 179CD 1C2          D1=D1- (oCPOSb)-(oDBEGb)-16     .
4101 179D0 141          DAT1=A A                        .
4102 179D3 03    RPLScc RTNCC
4103
4104            ********************************************************
4105            *
4106            * TFUUDL,TFUUD+  -  Update Line Number
4107            *
4108            * Entry:
4109            *       TFUUDL:
4110            *        D0     @  BCD line #
4111            *       TFUUD+:
4112            *        (OUTBS) @ BCD line #
4113            *
4114            * Exit:
4115            *       /LINE# =  line #
4116            *       Carry preserved
4117            *
4118            * Uses:  D0, A(A), C(A) (TFUUD+ only)
4119            *
4120 179D5 8E48  TFUUD+ GOSUBL dO=obs          D0 @ line #
          8F
4121 179DB D0    TFUUDL A=0    A               Read line #
4122 179DD 15A3         A=DATO 4                .
4123 179E1 1B4E         D0=(5) /LINE#          Store line #
          8F2
4124 179E8 140          DATO=A A                .
4125 179EB 01           RTN
```

```
4126
4127            ************************************************************
4128            *
4129            * TFUSLL  -  Store Line Length
4130            *
4131            * Entry:
4132            *       C(A)    =  Line length
4133            *
4134            * Exit:
4135            *       D0      @  /LNLEN
4136            *       C(A)    =  Line length
4137            *       C(S)    =  0
4138            *       Carry preserved
4139            *
4140            * Uses:  D0, C(S)
4141            *
4142 179ED 1B5F  TFUSLL D0=(5) /LNLEN         Store line length
          8F2
4143 179F4 144          DAT0=C A               .
4144 179F7 AC2          C=0    S
4145 179FA 01           RTN
4146
4147
4148            ***
4149            *
4150            *  Name:  OBAPND
4151            *
4152            *  Entry:
4153            *       C(A)    =  #nibs to write
4154            *
4155            *  Exit:
4156            *       P       =  0
4157            *       D0      @  Old av mem start
4158            *       A(A)    =  New av mem st
4159            *       D1      =  AVMEMS
4160            *
4161 179FC 20    OBAPND P=      0               Check memory with leeway
4162 179FE 8F00         GOSBVL =MEMCKL          .
          000
4163 17A05 400          RTNC
4164 17A08 130          D0=A                   D0 =  Av mem st
4165 17A0B C0           A=A+B  A               Update AVMEMS
4166 17A0D 141          DAT1=A A               .
4167 17A10 03           RTNCC
```

```
4168            SKIPLN  STITLE Skip Line Number
4169            ********************************************************************
4170            ********************************************************************
4171            **
4172            ** Name:    SKIPLN  -  Skip Line Number
4173            **
4174            ** Category:   PARUTL
4175            **
4176            ** Purpose:
4177            **      Skips a line number inan ASCII line.  Leading blanks,
4178            **      if present, are also skipped.  Scan also stops on EOL.
4179            **
4180            ** Entry:
4181            **      D1      @  Next character position in line
4182            **      P       =  0
4183            **
4184            ** Exit:
4185            **      Carry   =  Clear if line number found
4186            **      D1      @  New character position in line
4187            **      A(3-0)  =  Line number if carry clear
4188            **      P       =  0
4189            **
4190            ** Calls:      LIN#P
4191            **
4192            ** Uses.......
4193            **  Inclusive: A-D,R0, STMTR0
4194            **
4195            ** Stk lvls:   3
4196            **
4197            ********************************************************************
4198            ********************************************************************
4199 17A12 1B17  SKIPLN D0=(5) =S-R0-0        Set up fake output buffer
           8F2
4200 17A19 D3           D=0    A              Set up fake end of output buff
4201 17A1B CF           D=D-1  A               .
4202 17A1D 8D00         GOVLNG =LIN#P         Parse out line number
           000
```

```
4203            SWPBYT  STITLE Swap Bytes
4204            ********************************************************************
4205            ********************************************************************
4206            **
4207            ** Name:(S) SWPBYT  -  Swap Bytes
4208            **
4209            ** Category:   FILUTL
4210            **
4211            ** Purpose:
4212            **      Reverses A(3-2) and A(1-0).
4213            **
4214            ** Entry:
4215            **      A(3-0) =  2 bytes to be reversed
4216            **
4217            ** Exit:
4218            **      A(3-0) =  Reversed bytes
4219            **
4220            ** Calls:      None
4221            **
4222            ** Uses.......
4223            **  Inclusive: A(A),C(A)
4224            **
4225            ** Stk lvls:   0
4226            **
4227            ** History:
4228            **
4229            **    Date      Programmer              Modification
4230            **  --------    ----------   ----------------------------------
4231            **  09/21/82      FH         Designed and coded
4232            **
4233            ********************************************************************
4234            ********************************************************************
4235 17A24 D6   =SWPBYT C=A     A              Fetch low byte into high position
4236 17A26 F0           ASL     A               .
4237 17A28 F0           ASL     A               .
4238 17A2A F6           CSR     A              Put high byte low
4239 17A2C F6           CSR     A               .
4240 17A2E AEA          A=C     B               .
4241 17A31 03    SWPBcc RTNCC
4242
4243            *
4244            *
4245 17A33              END
```

```
 /COPYC  Abs  194773 #2F8D5 -   427   689  1035
 /DFIB#  Abs  194763 #2F8CB -   424   599   805  1900
 /DFTYP  Abs  194769 #2F8D1 -   426   101   642  1035  1092
 /DLEN   Abs  194778 #2F8DA -   431   489   823   488   822   877  3588
 /ERRCD  Abs  194757 #2F8C5 -   422   487   486  1054  1236
 /FLAG   Abs  194810 #2F8FA -   440  3366  3424
 /LINE#  Abs  194788 #2F8E4 -   433   489   823  1677  2326  4123
 /LNLEN  Abs  194805 #2F8F5 -   439  1219  4142
 /NSTAT  Abs  194699 #2F88B -  2892
 /NUMLN  Abs  194783 #2F8DF -   432   832  1677  2331
 /OPTN   Abs  194793 #2F8E9 -   435   484   483
 /PARM1  Abs  194795 #2F8EB -   436   788   791  3591
 /PARM2  Abs  194800 #2F8F0 -   437   484   791
 /SFIB#  Abs  194761 #2F8C9 -   423   523   599   738  1905
 /SFTYP  Abs  194765 #2F8CD -   425   487   530   642
 /STAT   Abs  194774 #2F8D6 -   429  1269  2349  2411
=?PRFI+  Abs   95104 #17380 -  2517   602
=?PRFIL  Abs   95102 #1737E -  2516   532
 ?PRFcc  Abs   95127 #17397 -  2524
 ALINFO  Ext                -    73
 AUTINC  Abs  194251 #2F6CB -    12  3385
 AVMEMS  Abs  193940 #2F594 -    12  3206  3377  3441  3763  3847
 B->C20  Abs   96649 #17989 -  4006  4000
 B->CHT  Abs   96632 #17978 -  3999  3737
 B->T70  Abs   96497 #178F1 -  3818  3807
 B->T80  Abs   96575 #1793F -  3846  3840
=B->TXT  Abs   96382 #1787E -  3744  1625  3354
 BBCOLL  Ext                -  1903
 BsErr   Ext                -   120
=CARYSV  Abs   94411 #170CB -  1740   145  1267  3002  3411
 CHAIN-  Ext                -  3614
 CLOSEF  Ext                -  1074
 COPYu   Ext                -   661
 CRTF    Ext                -   794
 CSLC5   Ext                -  2325
 CVUCW   Ext                -   709
 DO=/B+  Abs   94874 #1729A -  2264   699
 DO=/BX  Abs   94876 #1729C -  2265   705   734  2981  3043  3141
 DO=OBS  Ext                -  2250
 D1=CRS  Ext                -   717
 D1@AVS  Ext                -  3254
 D1@CPS  Abs   94711 #171F7 -  2170  1834
=DE-REF  Abs   94424 #170D8 -  1787   735
 DEFFIL  Ext                -   122
 EDITWF  Ext                -   733
 ENDALL  Ext                -   155
=EOFCHK  Abs   94463 #170FF -  1834  2168  2236
 EOLSN7  Ext                -  2720
 Eeofil  Abs   94807 #17257 -  2241  2238
 FFIB#   Ext                -  2183
=FIB#RS  Abs   94525 #1713D -  1911  2181
=FIB#SV  Abs   94516 #17134 -  1908
 FIBUPD  Abs   95465 #174E9 -  2904  2726  2998  3135
 FIBUcc  Abs   95501 #1750D -  2917  2910  2912
 FILEF   Ext                -   713
```

```
 FILFIL  Ext                  -   492
 FILSK+  Ext                  -  1788  2659
 FINDFS  Ext                  -   768
 FIXSPC  Ext                  -   166  2828  3149
 FLDEVX  Ext                  -   125   595
 FPOLL   Ext                  -  1602
 GOSUBI  Abs   94407 #170C7   -  1701   990
 GSBSTK  Abs  193955 #2F5A3   -    12  1197
 GTPTRX  Ext                  -  2233
=HEXASC  Abs   94536 #17148   -  1953  1966  3829
 HXASC3  Abs   94563 #17163   -  1963  1959
 HXASC5  Abs   94567 #17167   -  1964  1961
 INBS    Abs  194246 #2F6C6   -    12  3439
 LDCEXT  Ext                  -  3788
 LIF>20  Abs   94601 #17189   -  2032  2030
=LIF>NB  Abs   94581 #17175   -  2023  2813  3839
 LIN#P   Ext                  -  4202
 LNOVF?  Abs   96601 #17959   -  3896  3362
 LNPEXT  Ext                  -  3410
 LOCF05  Abs   94649 #171B9   -  2149  2187
 LOCF10  Abs   94680 #171D8   -  2159
 LOCF20  Abs   94704 #171F0   -  2167  2160  2163
=LOCFH*  Abs   94622 #1719E   -  2086  2993
 LOCFH-  Abs   94624 #171A0   -  2087  2658
=LOCFHD  Abs   94618 #1719A   -  2085  1031  3611
=LOCFI#  Abs   94745 #17219   -  2181  1134  2085  2573  2650  2730  2905  2985
                              4083  4094
=LOCFIL  Abs   94749 #1721D   -  2183   525   601
 LOCFcc  Abs   94743 #17217   -  2179  2240
 MEMCKL  Ext                  -  1674  4162
 MEMCL+  Ext                  -  3203
 MFERR*  Ext                  -   148
 MFERRS  Ext                  -   156
 MFWRNQ  Ext                  -  2341
 MOVE*M  Ext                  -  3214
 MVMEM+  Ext                  -  2663
 NIBL30  Abs   95593 #17569   -  2997  3032
 NIBL35  Abs   95622 #17586   -  3016  2992
 NIBL40  Abs   95628 #1758C   -  3017  2990
 NIBL50  Abs   95658 #175AA   -  3040  3028
 NIBL55  Abs   95699 #175D3   -  3065  3048
 NIBL57  Abs   95735 #175F7   -  3078  3072  3075
 NIBL60  Abs   95752 #17608   -  3093  3082
 NIBL64  Abs   95760 #17610   -  3097  3099
 NIBL65  Abs   95766 #17616   -  3098  3095
 NIBL66  Abs   95774 #1761E   -  3102  3106
 NIBL67  Abs   95786 #1762A   -  3105  3094
 NIBL68  Abs   95791 #1762F   -  3116  3100
 NIBL69  Abs   95811 #17643   -  3123  3119
 NIBL70  Abs   95818 #1764A   -  3125  3117  3122
 NIBL90  Abs   95840 #17660   -  3136  3132
 NIBL95  Abs   95859 #17673   -  3146  3137
 NIBLIO  Abs   95541 #17535   -  2981  2929
 NXtstm  Ext                  -   154
 OBAPND  Abs   96764 #179FC   -  4161  2926  3641
```

```
 OBCOLL  Ext                    -  3438
 OBED20  Abs   95904 #176A0  -  3206  3201
=OBEDIT  Abs   95879 #17687  -  3198  3052  3145  3463  3824
 OBLCMP  Ext                    -  2251
 OBPRD   Ext                    -  2252
 OPENF*  Ext                    -  1181
 OUTBS   Abs  193935 #2F58F  -    12  3377  3441  3763  3847
=OVERCK  Abs   94779 #1723B  -  2232  3019
 POLL    Ext                    -  2476
 POPUPD  Ext                    -   657  1022
 PRGFMF  Ext                    -  2474
 PSHUPD  Ext                    -   491
 PUGFIB  Ext                    -  1795
 PURG20  Abs   95092 #17374  -  2476  2470
=PURGEF  Abs   95065 #17359  -  2468  2465
 R<Rstk  Ext                    -  2262
 R<rstk  Abs   94866 #17292  -  2261   490  2984  3387
 RDBA20  Abs   95290 #1743A  -  2730  2754
 RDBA40  Abs   95311 #1744F  -  2737  2717
 RDBA60  Abs   95318 #17456  -  2739  2751
=RDBAS   Abs   95231 #173FF  -  2710  3744
 RDBAcc  Abs   95309 #1744D  -  2735
 RDBYTA  Ext                    -  3102
 RDINFO  Ext                    -  2258
 RDTE20  Abs   95436 #174CC  -  2823  2811  2819
=RDTEXT  Abs   95369 #17489  -  2801  3359
 READN2  Abs   95505 #17511  -  2920  2739
 READN4  Abs   95503 #1750F  -  2919  2737  2802
=READNB  Abs   95512 #17518  -  2924  2744  2814
 READWP  Abs   95507 #17513  -  2921
 RECAL-  Abs   95937 #176C1  -  3254  2748
 RECNIB  Abs   95934 #176BE  -  3253  2741  2804
 REWIND  Ext                    -  2575
 RFAD-I  Ext                    -  1794
 RLINFO  Ext                    -   675
 RPLLIN  Ext                    -  2995
 RPLS20  Abs   96693 #179B5  -  4092  4087  4089
=RPLSBH  Abs   96667 #1799B  -  4083  3609
 RPLScc  Abs   96723 #179D3  -  4102  3902  4007  4009  4096
=RRADD   Abs   95163 #173BB  -  2608   989
 RSTKR+  Abs   95600 #17570  -  3000  3018  3146
 RSTKR-  Abs   95602 #17572  -  3001  2986  2996  3026  3053
 RSTKRT  Abs   95604 #17574  -  3002
 Rstk<R  Ext                    -  2260
 Rstk<r  Abs   94858 #1728A  -  2259   656  1021  3004  3417
 S-RO-0  Abs  194673 #2F871  -    12  4199
 S-RO-1  Abs  194678 #2F876  -    12  1791
 S-R1-1  Abs  194694 #2F886  -    12  2265
 S-R1-2  Abs  194699 #2F88B  -    12  2892
 SETLEN  Abs   96262 #17806  -  3588  4001
 SETWRT  Ext                    -  3016
 SKIPLN  Abs   96786 #17A12  -  4199  3452  3800
 STAT40  Abs   95033 #17339  -  2416  2414
=STATR+  Abs   94988 #1730C  -  2361  1270
=STATRS  Abs   94963 #172F3  -  2351  2362
```

```
=STATSV  Abs   95023 #1732F -  2412
 STATcc  Abs   94986 #1730A -  2359
 STMTD1  Abs  194710 #2F896 -    12  2170  2185
 STMTR0  Abs  194673 #2F871 -    12  2711  2731
 SVINF0  Ext                 -  2147
=SWPBYT  Abs   96804 #17A24 -  4235  2812  3846
 SWPBcc  Abs   96817 #17A31 -  4241
 TFHD20  Abs   94266 #1703A -  1594  1592
 TFHD40  Abs   94306 #17062 -  1614  1611
 TFHD60  Abs   94349 #1708D -  1626  1606  1622
=TFHDLR  Abs   94255 #1702F -  1590   644
 TFLI20  Abs   94380 #170AC -  1674  1672
 TFLINE  Abs   94356 #17094 -  1667   844
 TFU020  Abs   93136 #16BD0 -   521   496
 TFU040  Abs   93047 #16B77 -   163   550
 TFU100  Abs   93270 #16C56 -   597   580   593
 TFU120  Abs   93294 #16C6E -   614   582   588   596
 TFU140  Abs   93325 #16C8D -   640   616   618   620
 TFU180  Abs   93360 #16CB0 -   656   647
 TFU190  Abs   93387 #16CCB -   673   660
 TFU200  Abs   93400 #16CD8 -   687   645
 TFU210  Abs   93427 #16CF3 -   694   692
 TFU220  Abs   93498 #16D3A -   716   711
 TFU230  Abs   93528 #16D58 -   726   721
 TFU240  Abs   93542 #16D66 -   732   723
 TFU250  Abs   93552 #16D70 -   734   719
 TFU260  Abs   93585 #16D91 -   759   693
 TFU270  Abs   93601 #16DA1 -   767   761
 TFU280  Abs   93640 #16DC8 -   783   769
 TFU290  Abs   93705 #16E09 -   805   803
 TFU300  Abs   93714 #16E12 -   821   744
 TFU310  Abs   93731 #16E23 -   832   882   910   961
 TFU315  Abs   93757 #16E3D -   841   836
 TFU320  Abs   93787 #16E5B -   874   869   935
 TFU330  Abs   93820 #16E7C -   895   875
 TFU340  Abs   93836 #16E8C -   900   898
 TFU350  Abs   93854 #16E9E -   927   872
 TFU360  Abs   93875 #16EB3 -   933   873
 TFU370  Abs   93915 #16EDB -   980   883
 TFU380  Abs   93937 #16EF1 -   987   982   985
 TFU385  Abs   93968 #16F10 -   996   804   992
 TFU390  Abs   93972 #16F14 -  1000   953   955
 TFU400  Abs   93976 #16F18 -  1021   514   930   994
 TFU420  Abs   94042 #16F5A -  1040  1028  1044
 TFU435  Abs   93633 #16DC1 -   781   995
 TFU440  Abs   94046 #16F5E -  1041  1026  1030  1039
 TFUCCK  Abs   94121 #16FA9 -  1135  3361
 TFUCLF  Abs   94088 #16F88 -  1073  1040
 TFUCLS  Abs   94084 #16F84 -  1072   658   704
 TFUDT-  Abs   94109 #16F9D -  1093
 TFUDTP  Abs   94102 #16F96 -  1092   640   785  1033
 TFUE20  Abs   94131 #16FB3 -  1139  1135
 TFUE0F  Abs   94117 #16FA5 -  1134  2710  2801
 TFUER*  Abs   93126 #16BC6 -   512   522   533
 TFUER2  Abs   93118 #16BBE -   510   494
```

```
 TFUER3  Abs   93315 #16C83 -   625   698
 TFUER4  Abs   93350 #16CA6 -   652   695
 TFUER5  Abs   93615 #16DAF -   774   714
 TFUER6  Abs   93621 #16DB5 -   775   737   795
 TFUERR  Abs   93128 #16BC8 -   513   626   778
 TFUOPN  Abs   94142 #16FBE - 1177   797
 TFUOPS  Abs   94139 #16FBB - 1176   521   736
 TFUPGD  Abs   95043 #17343 - 2461  1000
 TFURD+  Abs   94486 #17116 - 1899   896  1041
 TFURD-  Abs   94513 #17131 - 1906  1901
 TFURDC  Abs   94499 #17123 - 1903   980  1668
 TFURDD  Abs   94488 #17118 - 1900   983
 TFURDS  Abs   94506 #1712A - 1905   598  1024  1072  2084  2571
 TFURHA  Abs   94161 #16FD1 - 1197   687   959  1680  2602
 TFURIA  Abs   95153 #173B1 - 2602   694
 TFURLI  Abs   93387 #16CCB -  674   119  1023
 TFURLL  Abs   94181 #16FE5 - 1219   895
 TFURWD  Abs   95133 #1739D - 2572   987
 TFURWS  Abs   95129 #17399 - 2571   958
 TFURcc  Abs   94194 #16FF2 - 1222  1137
 TFUSEC  Abs   94196 #16FF4 - 1236   513   928   996
 TFUSFH  Abs   94614 #17196 - 2084   696
 TFUSLL  Abs   96749 #179ED - 4142  3360  3745
 TFUSTR  Abs   94956 #172EC - 2349   726   740   776  1075  2576
 TFUSTS  Abs   95016 #17328 - 2411   731   767   782   900  1073  1667  2319
                             2461  2572
 TFUSV1  Abs   94247 #17027 - 1274  1272
 TFUSVE  Abs   94212 #17004 - 1264   799   848   906   991  2466
 TFUUD+  Abs   96725 #179D5 - 4120  3422
 TFUUDL  Abs   96731 #179DB - 4121  3768
 TFUW20  Abs   94927 #172CF - 2334  2330
 TFUWRN  Abs   94888 #172A8 - 2319   933
 TFX100  Abs   92869 #16AC5 -   84   132
 TFX120  Abs   92900 #16AE4 -   99
 TFX200  Abs   92935 #16B07 -  116    89
 TFX220  Abs   92944 #16B10 -  119   127
 TFX225  Abs   92954 #16B1A -  122    91
 TFX230  Abs   92964 #16B24 -  125   117
 TFX240  Abs   92978 #16B32 -  128   108   123
 TFX300  Abs   92993 #16B41 -  144   130
 TFX320  Abs   93014 #16B56 -  149   147
=TRCFI#  Abs   95184 #173D0 - 2650   986
 TRFMBF  Abs  194757 #2F8C5 -   12   422   423   424   425   426   427   429
                             431   432   433   435   436   437   439   440
=TRSFM+  Abs   93057 #16B81 -  482   144
 TRSFMD  Ext                 -   66
 TRSFMP  Ext                 -   67
=TRSFMX  Abs   92856 #16AB8 -   73
=TRSFMu  Abs   93060 #16B84 -  483
 TXT-20  Abs   95989 #176F5 - 3366  3363
 TXT-40  Abs   96059 #1773B - 3415  3413
 TXT-60  Abs   96117 #17775 - 3437  3428
 TXT-80  Abs   96165 #177A5 - 3458  3456
=TXT->B  Abs   95965 #176DD - 3359  1620  3738
 TXTC20  Abs   96284 #1781C - 3595  3586
```

```
 TXTC40  Abs   96302 #1782E -  3602  3600
 TXTCHB  Abs   96257 #17801 -  3585  3353
 TXTCcc  Abs   96335 #1784F -  3615  3613
 UPCPOS  Ext                -  3130
 WRBYTD  Ext                -  3097
=WRITNB  Abs   95531 #1752B -  2931   904  4092
 WTAVFF  Abs   96337 #17851 -  3634  3748
 WTAVN2  Abs   96343 #17857 -  3637  3374  3480  3843
 WTAVN4  Abs   96341 #17855 -  3636
 WTAVWP  Abs   96345 #17859 -  3638  3603
 WTAVcc  Abs   96370 #17872 -  3646
 ZERPGM  Ext                -   722
 ^FU480  Abs   93581 #16D8D -   744   763
 ^FUERR  Abs   93321 #16C89 -   626   603   653
 ^^UERR  Abs   93629 #16DBD -   778
 bserr   Abs   92948 #16B14 -   120    74
=d0=obs  Abs   94815 #1725F -  2250  2752  3458  3466  3819  3830  4120
 eEOFIL  Ext                -  2241
 eFACCS  Ext                -   625
 eFEXST  Ext                -   774
 eFPROT  Ext                -  2521
 eFSPEC  Ext                -   126   511
 eFnFND  Ext                -  2188
 eILTFM  Ext                -   652
 eL2LNG  Ext                -  3426  3904
 eSYNTX  Ext                -  1058  3414
 eTFFLD  Ext                -   931
 eTFWRN  Ext                -  2339
 endall  Abs   93033 #16B69 -   155   150
 fBASIC  Abs   57876 #0E214 -    11  1617
 findf   Ext                -  2471
 fspecx  Ext                -   116
 lDBEGb  Abs      11 #0000B -    11  2175
 lFBEGb  Abs       6 #00006 -    11  2089
=mferrs  Abs   93040 #16B70 -   156   153
 move*m  Abs   95927 #176B7 -  3214  3031
 oCPOSb  Abs      40 #00028 -    11  1835  2177  2652  2913  4100
 oDBEGb  Abs      21 #00015 -    11  2157  2174  2177  4100
 oDEVCb  Abs      12 #0000C -    11  2153  2157
 oDLENb  Abs      46 #0002E -    11  1835  2652  2913  3040
 oFBEGb  Abs      13 #0000D -    11  2088
 oFLAGh  Abs      20 #00014 -    11  1037
 oFLSTr  Abs      49 #00031 -    11  4011
 oFTYPb  Abs       5 #00005 -    11   528
 oFTYPh  Abs      16 #00010 -    11  1032  1037
 oPROTb  Abs       9 #00009 -    11  2151  2153
 oRECLb  Abs      36 #00024 -    11  3040
 oSUBLn  Abs      37 #00025 -    11  4011
 oblcmp  Abs   94822 #17266 -  2251   876  2932  3469  3837  3899  4090
 obprd   Abs   94829 #1726D -  2252  2816  3758  3796
 pPURGE  Abs      16 #00010 -    11  2477
 pTRFMx  Abs      60 #0003C -    11  1603
=r<rstk  Abs   94868 #17294 -  2262
 rdinfd  Abs   94848 #17280 -  2257   578   759   783  2254
 rdinfo  Abs   94851 #17283 -  2258  1177  2256
```

```
 rdinfs  Abs    94841 #17279 -  2255   537   707
 rdinfx  Abs    94836 #17274 -  2253   614  2462
=rstk<r  Abs    94860 #1728C -  2260
 sCARD   Abs        2 #00002 -    11   493
 sDEST   Abs        3 #00003 -    11    75    90   129   131   796  1176  2255
                                2257
 sDRYRN  Abs        9 #00009 -   458   762   781   874   929   981   993  3585
                                3999
 sEOF    Abs        7 #00007 -    11   905  1136  1268  1271  1837  1840  2734
                                2806  3416  3747  3789
=sI/OBF  Abs       10 #0000A -   459   697  2158  2164  2654  2716  2911  2991
                                3027  3612  4088  4095
 sODD    Abs        8 #00008 -  2020  2033  2035  2818
 sODDEN  Abs        7 #00007 -  2889  3078  3083  3116  3131
 sODDST  Abs        6 #00006 -  2888  3045  3049  3071  3136
 sPRGCF  Abs       11 #0000B -   460   149   729
 sREAD   Abs        9 #00009 -  2890  2725  2909  2924  2931  2989  3074  3093
                                3118
 sTFEND  Abs        8 #00008 -   457   824   842   881   957  1273
 sTFERR  Abs        1 #00001 -   448  1029  1056  1238
 sTFINP  Abs        5 #00005 -   454   597   659   691   897   954  1027  2253
                                4006  4086
 sTFINV  Abs        6 #00006 -   455   835   952   956   984  1671  3599  4008
 sTFREQ  Abs        0 #00000 -   446   646  1590  1593  1626
 sTFWNG  Abs        2 #00002 -   450   870   927   934  1059
 sUNDEF  Abs        1 #00001 -    11   495
 svinf+  Abs    94639 #171AF -  2146   128
 svinfo  Abs    94642 #171B2 -  2147   165   712
 t!      Ext          -       -  3475
 tEOL    Ext          -       -  2719  2749  3478
 tINTO   Abs  3015151 #E01EF -    11    85
 tfu370  Abs    93816 #16E78 -   883   840   843
```

Input Parameters

  Source file name is FH&TFM::MS

  Listing file name is FH/TFM:TI:ML::-1

  Object file name is FH%TFM:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    M   M  N   N   &     FFFFF  RRRR   PPPP
 2            *    MM MM  N   N  & &    F      R   R  P   P
 3            *    M M M  NN  N  & &    F      R   R  P   P
 4            *    M M M  N N N   &     FFFF   RRRR   PPPP
 5            *    M   M  N  NN  & & &  F      R R    P
 6            *    M   M  N   N  &  &   F      R  R   P
 7            *    M   M  N   N   && &  F      R   R  P
 8            *
 9                TITLE  FREE/CLAIM PORT <831212.1205>
10 17A33         ABS    #17A33
11               RDSYMB TIXEQU::MS
12           ********************************************************************
13           ********************************************************************
14           **
15           ** Name:    FRPORT  -  FREE PORT Execute
16           **
17           ** Category:   STEXEC
18           **
19           ** Purpose:
20           **      Turn system RAM into independent RAM.
21           **
22           ** Entry:
23           **      D0 = PC.
24           **      Jumped on FREE token (PORT not tokenized).
25           **
26           ** Exit:
27           **      Through NXTSTM if no error.
28           **      =eDVCNF if port.dev is not system RAM.
29           **      =eMEM if insufficient memory.
30           **
31           ** Calls:      CNFFND, CONFS3, FINDWF, FNDDEV, MEMCKL, MOVED3,
32           **             MOVEU3, PDEV+, WIPOUT.
33           **
34           ** NOTE:
35           **      Will not execute if amount of available memory is less
36           **      than (module size + leeway).
37           **
38           ** Detail:
39           **      FREE PORT (P.DD).
40           **
41           ** Algorithm:
42           **      Search RAM table for requested port-device; eDVCNF if
43           **        not found.
44           **      Perform memory check with leeway + (if workfile does
45           **        not exist then oFLSTr else 0).  eMEM if not enough
46           **        memory to perform FREEPORT.
47           **      Write zero to size nibble of configuration table entry
48           **        for this device.  This informs the configuration
49           **        code that the device was intentionally removed and
50           **        machine should not coldstart.
51           **      If start-of-module >= (AVMEMS) then goto 1.
52           **      Source=(AVMEMS); Dest=(AVMEMS)+modsize; #nibs to move=
53           **        (AVMEMS)-(start-of-module); MOVED3 (move to higher
54           **        memory).  Goto 2.
55           **   1: If end-of-module <= (AVMEME) then goto 2.
```

```
                **      Source=(AVMEME); Dest=(AVMEME)-modsize; #nibs to move=
                **        (end-of-module)-(AVMEME); MOVEU3 (move to lower
                **        memory.
                **    2: Write IRAM ID (B3DDDDDE) to start of module.
                **       Zero out rest of module.
                **       Reconfigure system (call CONFS3).
                **       Exit through NXTSTM.
                **
                ** History:
                **
                **     Date      Programmer            Modification
                **  --------   ----------   ----------------------------------
                **  06/28/82   NM           Added documentation
                **
                ********************************************************************
                ********************************************************************
17A33 00000         REL(5) =FREEp
      0
17A38       =FRPORT
17A38 8F00          GOSBVL =PDEV+         Extract port,dev# from arg
      000
17A3F 20            P=     0
17A41 31FF          LCHEX  FF
17A45 8E00          GOSUBL =CNFFND        Look for RAMtable
      00
17A4B 460           GOC    FRPO20         Go if found
17A4E 68D0          GOTO   SYSERR         System error
17A52 AB8   FRPO20  B=A    X              Hold size in B
17A55 7621          GOSUB  FNDDEV         Find this pdev# in table
17A59 102           R2=A                  Stash module address
17A5C D4            A=B    A
17A5E 100           R0=A                  Stash module size
17A61 171           D1=D1+ 2              Point at size nibble
17A64 137           CD1EX                 Retrieve pointer to size nibble
17A67 109           R1=C                  Stash for later
17A6A 8F00          GOSBVL =FINDWF        Look for workfile
      000
17A71 D2            C=0    A
17A73 560           GONC   YESHWF         It exists
17A76 3113          LC(2)  =oFLSTr        Doesn't exist.  Must ensure room.
17A7A 110   YESHWF  A=R0
17A7D C2            C=A+C  A
17A7F 8F00          GOSBVL =MEMCKL        AVMEM check w/LEEWAY
      000
17A86 560           GONC   FRPO70         Go if sufficient memory
17A89 6031          GOTO   INSMEM         Insufficient memory
17A8D 119   FRPO70  C=R1                  Retrieve ptr to size nibble
17A90 134           D0=C                  Point at size nib for this entry
17A93 AC2           C=0    S
17A96 1544          DAT0=C S              Write 0 for orderly removal
17A9A D6            C=A    A              AVMEMS
17A9C 110           A=R0
17A9F D8            B=A    A              Retrieve module size
17AA1 112           A=R2                  Address of module start.
17AA4 8BE           ?A>=C  A              Mod starts before AVMEMS?
```

```
106 17AA7 81          GOYES  FRP120        No
107 17AA9 134         DO=C                 Yes, source
108 17AAC C9          C=C+B  A              Add module size for...
109 17AAE 135         D1=C                 Destination
110 17AB1 E9          C=C-B  A              AVMEMS
111 17AB3 E2          C=C-A  A              Sub modstart for #nibs to move
112 17AB5 8E00        GOSUBL =MOVED3       Move down (to higher address)
          00
113 17ABB 6130        GOTO   MODINT        Initialize module at DO
114 17ABF 1E00 FRP120 D1=(4) =AVMEME
          00
115 17AC5 147         C=DAT1 A              AVMEME
116 17AC8 130         DO=A                 Hold module start addr
117 17ACB C0          A=A+B  A              End of module address
118 17ACD 8BA         ?A<=C  A              Module ends before AVMEME?
119 17AD0 D1          GOYES  MODINT        Yes, ready to initialize
120 17AD2 134         DO=C                 Source
121 17AD5 E9          C=C-B  A              Subtract module size for...
122 17AD7 135         D1=C                 Destination
123 17ADA E0          A=A-B  A              Module starting address
124 17ADC 100         RO=A                 Hold
125 17ADF EE          C=A-C  A              #nibbles to move
126 17AE1 8E00        GOSUBL =MOVEU3       Move up (to lower address)
          00
127 17AE7 110         A=RO                 Retrieve start addr of module
128 17AEA 130         DO=A                 Stash in DO
129 17AED 20    MODINT P=     0            Ready to initialize module
130 17AEF AF2         C=0    W
131 17AF2 37B3        LCHEX  EDDDDD3B      IRAM id word
          DDDD
          DE
132 17AFC 1547        DAT0=C W             Initialize IRAM
133 17B00 D9          C=B    A
134 17B02 137         CD1EX
135 17B05 1CF         D1=D1- 16            Modsize - 16
136 17B08 136         CDOEX
137 17B0B 137         CD1EX
138 17B0E 17F         D1=D1+ 16            Mod addr + 16
139 17B11 8E00 FRPEND GOSUBL =WIPOUT       Clear out module
          00
140 17B17 840         ST=0   0             Setup for powerup configure
141 17B1A 853         ST=1   3             Force memory-changed config.
142 17B1D 8E00        GOSUBL =CONFS3       Do configuration.
          00
143 17B23 6000        GOTO   =READ95       GOTO NXTSTM
144           *
145 17B27 8D00 =SYSERR GOVLNG =CORUPT
          000
146           *
147           ********************************************************************
148           ********************************************************************
149           **
150           ** Name:    NASSAU  -  CLAIM PORT Execute
151           **
152           ** Category:   STEXEC
```

```
153                     **
154                     ** Purpose:
155                     **      Make an independent RAM a system RAM.
156                     **
157                     ** Entry:
158                     **      D0 = PC.
159                     **      Jumped on CLAIM token (PORT not tokenized).
160                     **
161                     ** Exit:
162                     **      Through NXTSTM.
163                     **      =eDVCNF if requested port.dev is not an IRAM.
164                     **
165                     ** Calls:       CLMPRO, CNFFND, FNDDEV, PDEV+, CONFS3, WIPOUT.
166                     **
167                     **
168                     ** NOTE:
169                     **      Does not execute if any secured files exist in the
170                     **      IRAM's file chain.
171                     **
172                     ** Detail:
173                     **      CLAIM PORT (P.DD).
174                     **
175                     ** Algorithm:
176                     **      Look for entry in ROM table, eDVCNF if not found.
177                     **      Check filechain for secure files; eFPROT if any.
178                     **      Clear module, including IRAM ID.
179                     **      Reconfigure system.
180                     **
181                     ** History:
182                     **
183                     **    Date      Programmer              Modification
184                     **  --------   ----------   ---------------------------------
185                     **  06/28/82   NM           Added documentation
186                     **
187                     ************************************************************
188                     ************************************************************
189 17B2E 0000 0         REL(5) =CLAIMp
190 17B33               =NASSAU
191 17B33 8F00 000      GOSBVL =PDEV+           Get port-device information
192 17B3A 20            P=     0
193 17B3C 31EF          LCHEX  FE               Rom buffer id
194 17B40 8E00 00       GOSUBL =CNFFND          Find it
195 17B46 50E           GONC   SYSERR
196 17B49 DC            ABEX   A                Swap buffersize with pdev
197 17B4B 7030          GOSUB  FNDDEV
198 17B4F 25            P=     5                Returned...device must exist
199 17B51 909           ?B=0   P                Is it RAM?
200 17B54 60            GOYES  NASS20           Yes
201 17B56 6760          GOTO   NOTRAM           No, error!
202 17B5A D9    NASS20  C=B    A
203 17B5C D7            D=C    A                Stash module size
204 17B5E 130           D0=A
```

```
205 17B61 131          D1=A
206 17B64 177          D1=D1+ 8              Point at 1st file in chain
207 17B67 137          CD1EX
208 17B6A 20           P=     0
209 17B6C 8F00         GOSBVL =CLMPRO        Check for secure files
          000
210 17B73 136          CD0EX                 There are none
211 17B76 135          D1=C                  Restore pointer to module
212 17B79 DB           C=D    A              Module size
213 17B7B 659F         GOTO   FRPEND         Reconfigure system
214               ****************************************************************
215               ****************************************************************
216               **
217               ** Name:    FNDDEV  -  Find Port, DevW Requested In Cnf Tbl
218               **
219               ** Category:   LOCAL
220               **
221               ** Purpose:
222               **      Locate a port/device in a configuration table.
223               **
224               ** Entry:
225               **      D1 points at first table entry.
226               **      B[X]=length of table.
227               **      D[B]=pdev looking for (port in D[1], dev in D[0]).
228               **
229               ** Exit:
230               **      =eDVCNF if not found.
231               **      Else B[A]=device size (chipsize * chipcount),
232               **           A[A]=address,
233               **           B[5]=device type.
234               **      D1 points at nibble 1 of table entry found.
235               **      P=0.
236               **
237               ** Calls:      MSIZE+ (falls through).
238               **
239               ** Uses.......
240               **             A,B,C,D1,P.
241               **
242               ** Stk lvls:   1
243               **
244               ** History:
245               **
246               **    Date     Programmer             Modification
247               **  --------   ----------   ---------------------------------
248               **  06/28/82   NM           Added documentation
249               **
250               ****************************************************************
251               ****************************************************************
252 17B7F 20    FNDDEV P=     0
253 17B81 1C8          D1=D1- 9
254 17B84 179   FNDD10 D1=D1+ 10
255 17B87 32A0         LCHEX  00A            Size of one entry
          0
256 17B8C B31          B=B-C  X              Any more entries?
257 17B8F 4E2          GOC    NODEV          No.  dev rqst'd not exist
```

```
258 17B92 14F   FNDD30 C=DAT1 B
259 17B95 9E7          ?C>D   B            At higher pdev in table?
260 17B98 62           GOYES  NODEV        Yes, no such device.
261 17B9A 9E3          ?C<D   B            At lower pdev in table?
262 17B9D 7E           GOYES  FNDD10       Yes, look some more
263 17B9F 15F7         C=DAT1 8            Read table entry.
264 17BA3 AF5          B=C    W            Addr and devtype to B
265 17BA6 BF5          BSR    W
266 17BA9 AE1          B=0    B            Clear garbage in address
267 17BAC 8E00         GOSUBL =MSIZE+      Size/100H to A
          00
268 17BB2 F0           ASL    A
269 17BB4 F0           ASL    A            Size in A
270 17BB6 DC           ABEX   A            Size to B, Address to A
271 17BB8 01           RTN
272             *
273 17BBA 6000  INSMEM GOTO   =MEMERj       GOTO MEMERR
274             *
275 17BBE       NOTRAM
276 17BBE 20    NODEV  P=     0
277 17BC0 8D00         GOVLNG =DVCNFE
          000
278             *
279 17BC7              END
```

AVMEME	Ext			-	114		
CLAIMp	Ext			-	189		
CLMPRO	Ext			-	209		
CNFFND	Ext			-	77	194	
CONFS3	Ext			-	142		
CORUPT	Ext			-	145		
DVCNFE	Ext			-	277		
FINDWF	Ext			-	88		
FNDD10	Abs	97156	#17B84	-	254	262	
FNDD30	Abs	97170	#17B92	-	258		
FNDDEV	Abs	97151	#17B7F	-	252	81	197
FREEp	Ext			-	72		
FRP070	Abs	96909	#17A8D	-	97	95	
FRP120	Abs	96959	#17ABF	-	114	106	
FRPEND	Abs	97041	#17B11	-	139	213	
FRP020	Abs	96850	#17A52	-	80	78	
=FRPORT	Abs	96824	#17A38	-	73		
INSMEM	Abs	97210	#17BBA	-	273	96	
MEMCKL	Ext			-	94		
MEMERj	Ext			-	273		
MODINT	Abs	97005	#17AED	-	129	113	119
MOVED3	Ext			-	112		
MOVEU3	Ext			-	126		
MSIZE+	Ext			-	267		
NASS20	Abs	97114	#17B5A	-	202	200	
=NASSAU	Abs	97075	#17B33	-	190		
NODEV	Abs	97214	#17BBE	-	276	257	260
NOTRAM	Abs	97214	#17BBE	-	275	201	
PDEV+	Ext			-	74	191	
READ95	Ext			-	143		
=SYSERR	Abs	97063	#17B27	-	145	79	195
WIPOUT	Ext			-	139		
YESHWF	Abs	96890	#17A7A	-	92	90	
oFLSTr	Abs	49	#00031	-	11	91	

Input Parameters

  Source file name is MN&FRP::MS

  Listing file name is MN/FRP:TI:ML::-1

  Object file name is MN%FRP:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS   BBBB    &     RRRR   DDDD
 2            *    S   S  B   B  & &    R   R   D  D
 3            *    S      B   B  & &    R   R   D  D
 4            *     SSS   BBBB    &     RRRR    D  D
 5            *        S  B   B  & & &  R R     D  D
 6            *    S   S  B   B  &  &   R  R    D  D
 7            *     SSS   BBBB    && &  R   R  DDDD
 8            *
 9                    TITLE  READ Statement Execute <831216.1605>
10 17BC7             ABS    #17BC7
11           ********************************************************************
12           ********************************************************************
13           **
14           ** Name:    READ    -  READ Statement Execute
15           **
16           ** Category:   STEXEC
17           **
18           ** Purpose:
19           **      Implements READ statement
20           **
21           ** Entry:
22           **      DO points past READ token
23           **      P       =  0
24           **
25           ** Exit:
26           **      Exits through NXTSTM
27           **
28           ** Calls:      TKSCN+,STKCHR,STKCH+,NXTSTM
29           **
30           ** Detail:
31           **      Checks if first byte is a # token and jumps into
32           **      READ# routine if it is.  Otherwise, for each
33           **      variable in READ list, the destination address and
34           **      type are determined.  The next data item seached for
35           **      as follows.  If DATPTR = 0 then it is set initialized
36           **      to PRGMST.  If the byte pointed to by DATPTR is not
37           **      a comma token then TOKSCN is used to find the next
38           **      DATA statement in the program.  If none is found then
39           **      a "No data" error is generated.  Otherwise,
40           **      the data is parsed according to the data type of the
41           **      destination.
42           **
43           **      Expression parse will be called on the data stream.
44           **      If a valid expression of matching type (string/num-
45           **      eric) is found and the next token is either a comma
46           **      of an EOL then the expression is moved onto stack
47           **      and evaluated.  If the destination is string and
48           **      either an invalid or numeric expression is found
49           **      or if the expression is not followed by a comma or
50           **      EOL then the pointer is backed up to the start of
51           **      the data field and the data item is parsed as
52           **      an unquoted string.
53           **      Before giving control to expression execute DATPTR
54           **      must be updated to point to either the comma
55           **      following the data or at an EOL.
```

```
56          **
57          ** History:
58          **
59          **    Date     Programmer              Modification
60          **  --------   ----------   ----------------------------------
61          **  10/25/83   B.S.         Updated documentation
62          **
63          ********************************************************************
64          ********************************************************************
65           InvalE EQU    0              Status bit definition
66           NumExp EQU    3              Status bit definition
67           StrReq EQU    5              Status bit definition
68          ********************************************************************
69          ********************************************************************
70          **
71          ** Name:(S) RDATTY  -  Report "Data Type" error
72          **
73          ** Category:   SYSTEM
74          **
75          ** Purpose:
76          **      To report "Data Type" as an execution error.
77          **
78          ** Entry:
79          **      S13=0 if program not running (i.e., keyboard
80          **        execution error)
81          **      S13=1 if running program
82          **      No other necessary conditions.
83          **
84          ** Exit:
85          **      Exits to BASIC main loop (ERRRTN)
86          **
87          ** Calls:      MFERR
88          **
89          ** Uses....... BASIC main loop can use anything
90          **
91          ** Stk lvls:   BASIC main loop can use anything
92          **
93          ** NOTE:
94          **      RDATTY sets P=0 to selec the following error options:
95          **        -- not a parse error
96          **        -- store ERRN (and ERRL if S13=1)
97          **        -- display "ERR:" (or ERR L<#>:")
98          **
99          ** Detail:
100         **     =RDATTY P=     0
101         **             LC(2)  =eDATTY
102         **             GOLONG =MFERR
103         **
104         ** History:
105         **
106         **    Date     Programmer              Modification
107         **  --------   ----------   ----------------------------------
108         **  11/09/83   MB           Documentation
109         **
110         ********************************************************************
```

```
111           ****************************************************************
112 17BC7 0000          REL(5) =READDC
          0
113 17BCC 0000          REL(5) =READP
          0
114 17BD1       =READ
115 17BD1 14A           A=DAT0 B
116 17BD4 3132          LC(2)  \#\
117 17BD8 966           ?A#C   B              Is it a # token
118 17BDB 80            GOYES  READ00         No, then do READ
119 17BDD 8C00          GOLONG =READ#         Yes, the handle READ# statement
          00
120             *-
121             *-
122 17BE3 136   READ00  CD0EX
123 17BE6 108           R0=C                  Save D0
124 17BE9 8F00          GOSBVL =PRSCKB        Set up program scope pointers
          000
125 17BF0 118           C=R0
126 17BF3 134           D0=C                  Restore D0
127 17BF6 8E00          GOSUBL =NXTVAR
          00
128 17BFC 8E00          GOSUBL =D1MST+
          00
129 17C02 8E00          GOSUBL =STKVCT
          00
130 17C08 845   READ03  ST=0   StrReq
131 17C0B 862           ?ST=0  2
132 17C0E 50            GOYES  READ05
133 17C10 855           ST=1   StrReq
134 17C13       READ05
135 17C13 861           ?ST=0  1              Is it an array?
136 17C16 C0            GOYES  READ07         No, then collapse stack
137 17C18 8E00          GOSUBL =NXTADR        Yes, then set up for first element
          00
138 17C1E 6700          GOTO   READ09
139             *-
140             *-
141 17C22 7000  READ07  GOSUB  =collap        Collapse MTHSTK to FORSTK
142 17C26 1B00  READ09  D0=(5) =PRGMEN
          000
143 17C2D 146           C=DAT0 A
144 17C30 D7            D=C    A              Copy PRGMEN to D(A)
145 17C32 1A00          D0=(4) =DATPTR
          00
146 17C38 146           C=DAT0 A              Read DATPTR
147 17C3B 8AE           ?C#0   A              Is DATPTR=0?
148 17C3E B0            GOYES  READ10         No, then okay
149 17C40 1A00          D0=(4) =PRGMST        Yes, then use PRGMST
          00
150 17C46 146           C=DAT0 A
151 17C49 134   READ10  D0=C
152 17C4C 14A           A=DAT0 B              Read token
153 17C4F 31C2          LCASC  \,\            ASCII comma
154 17C53 962           ?A=C   B              At comma?
```

```
155 17C56 F1          GOYES  READ20        Yes, then data follows
156 17C58 3100        LC(2)  =tDATA        No, then find next DATA
157 17C5C 161         D0=D0+ 2             Move past EOL or @
158 17C5F 90C         ?A#0   P             Is it an EOL?
159 17C62 50          GOYES  READ15        No, then don't need to skip line #
160 17C64 163         D0=D0+ 4             Skip over line #
161 17C67 8F00 READ15 GOSBVL =TKSCN4       Find next data statement
          000
162 17C6E 3100        LC(2)  =eNODAT       Error:  No data
163 17C72 595         GONC   MFERRj        No DATA found?  Then error
164 17C75 161  READ20 D0=D0+ 2             Move ptr to start of data
165 17C78 132         AD0EX
166 17C7B 102         R2=A                 Save start of data field
167 17C7E 131         D1=A                 D1=Start of data field
168 17C81 8F00        GOSBVL =D0=AVS       D0=(AVMEMS)
          000
169 17C88 103         R3=A                 R3 contains (AVMEMS)
170 17C8B 8F00        GOSBVL =EXPPAR       Parse expression
          000
171 17C92 D8          B=A    A             Save first token not used in expr
172 17C94 133         AD1EX
173 17C97 8E00        GOSUBL =D1=AVE
          00
174 17C9D 109         R1=C
175 17CA0 131         D1=A
176 17CA3 D4          A=B    A             Get first token not used in expr
177 17CA5 3100        LC(2)  =tCOMMA
178 17CA9 962         ?A=C   B             Is it a comma?
179 17CAC D0          GOYES  READ30        Yes, then okay
180 17CAE 300         LC(1)  =tEOL
181 17CB1 966         ?A#C   B             Is it an EOL?
182 17CB4 95          GOYES  READ59        No, then expr wasn't really valid
183 17CB6 173         D1=D1+ 4             Point at real EOL token + 2
184 17CB9 1C1  READ30 D1=D1- 2             Point at comma or EOL
185 17CBC 875         ?ST=1  StrReq        String data required?
186 17CBF 95          GOYES  READ65
187 17CC1 860  READ40 ?ST=0  InvalE        Valid expression?
188 17CC4 E0          GOYES  READ45        Yes, then skip
189 17CC6 20  =RDATTY P=     0
190 17CC8 3100        LC(2)  =eDATTY       Data Type error
191 17CCC 8C00 =MFERRj GOLONG =MfErr
          00
192           *-
193           *-
194 17CD2 863  READ45 ?ST=0  NumExp        Numeric expression found?
195 17CD5 1F          GOYES  RDATTY        No, then data type error
196 17CD7 8F00 READ50 GOSBVL =OUT1TK       Output either comma or EOL token
          000
197 17CDE 137         CD1EX
198 17CE1 1F00        D1=(5) =DATPTR
          000
199 17CE8 145         DAT1=C A             Save updated data pointer
200 17CEB 119         C=R1                 Recall (AVMEME)
201 17CEE 135         D1=C                 D1 points at end of av mem
202 17CF1 8E00        GOSUBL =AVMMOV       Move buffer onto stack before AVMEM
```

```
          00
203 17CF7 137          CD1EX                Move buffer start to C(A)
204 17CFA 135          D1=C                 Copy buffer start back to D1 (stack
205 17CFD 134          D0=C                 Copy buffer start to D0 (PC)
206 17D00 3497         LC(5)  READ90        Return to READ90 after EXPR and STO
          D71
207 17D07 8C00         GOLONG =EX-STC       Call EXPR and STORE then rtn to C(A
          00
208               *-
209               *-
210 17D0D 850   READ59 ST=1   InvalE        Expression wasn't really valid
211 17D10 865   READ60 ?ST=0  StrReq        Is a string required?
212 17D13 EA           GOYES  READ40        No, then data type error
213 17D15 570          GONC   READ70        (B.E.T.)  Skip
214               *-
215               *-
216 17D18 863   READ65 ?ST=0  NumExp        Numeric expression found?
217 17D1B CB           GOYES  READ50        No, then valid string must have bee
218 17D1D 11A   READ70 C=R2                 Recall start of data string
219 17D20 134          D0=C                 D0 points to start of string
220 17D23 119          C=R1                 Recall (AVMEME)
221 17D26 135          D1=C                 D1 is stack pointer
222 17D29 11B          C=R3                 Recall (AVMEMS)
223 17D2C D7           D=C    A             D(A)=(AVMEMS) (for STKCHR)
224 17D2E 14A   READ75 A=DAT0 B             Read a byte
225 17D31 31C2         LCASC  \,\           ASCII comma separates data items
226 17D35 962          ?A=C   B             Is it a CR?
227 17D38 71           GOYES  READ82        Yes, then exit loop
228 17D3A 161          D0=D0+ 2             Move data pointer to next char
229 17D3D 31D0         LC(2)  13            CR terminates line
230 17D41 962          ?A=C   B             Is it a comma?
231 17D44 B0           GOYES  READ82        Yes, then exit loop
232 17D46 D6           C=A    A             Copy byte to C(B) for output
233 17D48 7000         GOSUB  =STKCHR       Output char to stack
234 17D4C 51E          GONC   READ75        (B.E.T.)  Loop back
235               *-
236               *-
237 17D4F 136   READ82 CD0EX                C(A)=Moved data pointer
238 17D52 1B00         D0=(5) =DATPTR
          000
239 17D59 144          DAT0=C A             Write out updated data pointer
240 17D5C 850          ST=1   0             Want ADHEAD to return
241 17D5F 7000         GOSUB  =ADHEAD       Add string header to stack
242 17D63 8E00         GOSUBL =D1STOR       Store string at destination
          00
243 17D69 8E00         GOSUBL =D1=AVE
          00
244 17D6F 7000         GOSUB  =popnth       Pop STOREd string from stack
245 17D73 137          CD1EX                Put pointer in C(A)
246 17D76 580          GONC   READ91        (B.E.T.)
247               *-
248               *-
249 17D79 161   READ90 D0=D0+ 2             Skip past expression terminator
250 17D7C 136          CD0EX                Put pointer in C(A)
251 17D7F 8E00  READ91 GOSUBL =AVE=C        Reset MTHSTK
```

```
          00
  252 17D85 8E00          GOSUBL =CHKDON        Check if doing an array
          00
  253 17D8B 411           GOC    READ94         No, then do next var
  254 17D8E 08            CLRST                 Yes, then ...
  255 17D90 8E00          GOSUBL =STKVCT        ...Check if done with array
          00
  256 17D96 560           GONC   READ94         Yes, then do next var
  257 17D99 6E6E          GOTO   READ03         Loop back to read another item.
  258             *-
  259             *-
  260 17D9D 8E00  READ94  GOSUBL =RESTDO
          00
  261 17DA3 14A           A=DAT0 B              Read byte at program counter
  262 17DA6 161           D0=D0+ 2              Move pointer past possible comma to
  263 17DA9 3100          LC(2)  =tCOMMA
  264 17DAD 966           ?A#C   B              Is it a comma token?
  265 17DB0 60            GOYES  READ95         No, then done with READ stmt execut
  266 17DB2 603E          GOTO   READ00         Yes, then loop back for next read
  267             *-
  268             *-
  269 17DB6 8C00 =READ95  GOLONG =NXtstm        Continue with next statement
          00
  270 17DBC               END
```

ADHEAD	Ext			-	241				
AVE=C	Ext			-	251				
AVMMOV	Ext			-	202				
CHKDON	Ext			-	252				
DO=AVS	Ext			-	168				
D1=AVE	Ext			-	173	243			
D1MST+	Ext			-	128				
D1STOR	Ext			-	242				
DATPTR	Ext			-	145	198	238		
EX-STC	Ext			-	207				
EXPPAR	Ext			-	170				
InvalE	Abs	0	#00000	-	65	187	210		
=MFERRj	Abs	97484	#17CCC	-	191	163			
MfErr	Ext			-	191				
NXTADR	Ext			-	137				
NXTVAR	Ext			-	127				
NXtstm	Ext			-	269				
NumExp	Abs	3	#00003	-	66	194	216		
OUT1TK	Ext			-	196				
PRGMEN	Ext			-	142				
PRGMST	Ext			-	149				
PRSCKB	Ext			-	124				
=RDATTY	Abs	97478	#17CC6	-	189	195			
=READ	Abs	97233	#17BD1	-	114				
READ#	Ext			-	119				
READ00	Abs	97251	#17BE3	-	122	118	266		
READ03	Abs	97288	#17C08	-	130	257			
READ05	Abs	97299	#17C13	-	134	132			
READ07	Abs	97314	#17C22	-	141	136			
READ09	Abs	97318	#17C26	-	142	138			
READ10	Abs	97353	#17C49	-	151	148			
READ15	Abs	97383	#17C67	-	161	159			
READ20	Abs	97397	#17C75	-	164	155			
READ30	Abs	97465	#17CB9	-	184	179			
READ40	Abs	97473	#17CC1	-	187	212			
READ45	Abs	97490	#17CD2	-	194	188			
READ50	Abs	97495	#17CD7	-	196	217			
READ59	Abs	97549	#17D0D	-	210	182			
READ60	Abs	97552	#17D10	-	211				
READ65	Abs	97560	#17D18	-	216	186			
READ70	Abs	97565	#17D1D	-	218	213			
READ75	Abs	97582	#17D2E	-	224	234			
READ82	Abs	97615	#17D4F	-	237	227	231		
READ90	Abs	97657	#17D79	-	249	206			
READ91	Abs	97663	#17D7F	-	251	246			
READ94	Abs	97693	#17D9D	-	260	253	256		
=READ95	Abs	97718	#17DB6	-	269	265			
READDC	Ext			-	112				
READP	Ext			-	113				
RESTDO	Ext			-	260				
STKCHR	Ext			-	233				
STKVCT	Ext			-	129	255			
StrReq	Abs	5	#00005	-	67	130	133	185	211
TKSCN4	Ext			-	161				
collap	Ext			-	141				

eDATTY	Ext	-	190	
eNODAT	Ext	-	162	
popnth	Ext	-	244	
tCOMMA	Ext	-	177	263
tDATA	Ext	-	156	
tEOL	Ext	-	180	

Input Parameters

  Source file name is SB&RD::MS

  Listing file name is SB/RD:TI:ML::-1

  Object file name is SB%RD:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS   BBBB    &     III  00000
 2            *    S   S  B   B  & &     I   0   0
 3            *    S      B   B  & &     I   0   0
 4            *     SSS   BBBB    &      I   0   0
 5            *        S  B   B  & & &   I   0   0
 6            *    S   S  B   B  &  &    I   0   0
 7            *     SSS   BBBB    && &  III  00000
 8            *
 9                     TITLE  I/O Execution Routines <831228.1849>
10 17DBC              ABS    #17DBC
11                    RDSYMB TI%EQU::MS
12
13           ************************************************
14           *           Status Bit Definitions             *
15           ************************************************
16            Return EQU    0              Return vs GOLONG EXPR
17            Blanks EQU    1              Leading/Trailing spaces required
18            InvalE EQU    0              Invalid expression (parse)
19            String EQU    1              String expression (parse)
20            NumExp EQU    3              Numeric expression (parse)
21            NoCur  EQU    3              Display flag NoCur
22            InhEOL EQU    4              Need to inhibit ENDLINE string
23            S/NFlg EQU    5              String or Number on stack flag
24           *   The following are offsets from STMTRO to get
25           *     to specified field
26            oHNDLR EQU    1              Handler address (5 nib field)
27            oPOSIT EQU    6              POS/WIDTH address (5 nib field)
28            oEOLLN EQU    11             Length of EOL string
29                                           (1 nib field)
30            oEOLC1 EQU    12             First char of EOL string
31                                           (0..6 nib field)
```

```
                        STITLE SENDEL - Send end line to device
                *****************************************************************
                *****************************************************************
                **
                ** Name:(S) SENDEL  -  Send EndLine to Device via Handler
                **
                ** Category:   EXCUTL
                **
                ** Purpose:
                **      Transmit an "EndLine" to a device by calling the
                **      the appropriate handler routine.
                **      Updates column count by the number of characters
                **      in buffer.
                **
                ** Entry:
                **      Statement scratch set up by CKINFO
                **
                ** Exit:
                **      P = 0
                **
                ** Calls:      Device handler specified in statement scratch
                **
                ** Uses.......
                **  Exclusive: A(W), C(A), D(A), D1
                **  Inclusive: A(W), B(W), C(W), D(W), D1, P, R1(W), R2(A)
                **             Does not use D0, Status.
                **
                ** Stk lvls:   3
                **
                ** Note:  DO NOT USE D0 OR STATUS BITS!!!!
                **
                ** Detail:
                **      This routine calls the Part 2 handler by entering
                **      the SENDIT code.
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   ----------------------------------
                **  06/25/82   B.S.         Updated documentation
                **
                *****************************************************************
                *****************************************************************
17DBC 874    CRLFCK ?ST=1  InhEOL          CR/LF required?
17DBF 00            RTNYES                No, then return
17DC1 1F00  =SENDEL D1=(5) (=STMTR0)+oEOLLN
      000
17DC8 AF0           A=0     W
17DCB 15B0          A=DAT1  1             Read length
                *
17DCF 81C           ASRB                  (NZ) Convert to bytes
                *
17DD2 170           D1=D1+  1             Skip length
17DD5 137           CD1EX
17DD8 D7            D=C     A             D(A)=Start of buffer
```

```
     86 17DDA 7323          GOSUB  D1@POS
     87 17DDE D2            C=0    A
     88 17DE0 532           GONC   SEND30          B.E.T.
```

```
                STITLE SENDIT - Send buffer to device
*****************************************************************
*****************************************************************
**
** Name:(S) SENDIT  -  Send Buffer to Device via Handler
** Name:(S) SEND20  -  Send Buffer to Device via Handler
**
** Category:   EXCUTL
**
** Purpose:
**      Transmit a buffer of 8-bit ASCII characters to a
**      device by calling the appropriate handler routine.
**      Updates column count by the number of characters
**      in buffer.
**
** Entry:
**      Statement scratch set up by CKINFO
**      SENDIT:
**        D1 points to first byte of buffer
**        Buffer end is at (AVMEME)
**      SEND20:
**        D(A) point to first byt of buffer
**        A(A) is length of buffer (in bytes)
**
** Exit:
**      P = 0
**
** Calls:       D1@POS,
**              Device handler specified in statement scratch
**
** Uses.......
**  Exclusive: A(W), C(A), D(A), D1
**  Inclusive: A(W), B(W), C(W), D(W), D1, P, R1(W), R2(A)
**              Does not use D0, Status.
**
** Stk lvls:   <4
**
** Note: DO NOT CHANGE D0 OR STATUS BITS!!!!
**
** Detail:
**      For the IO handler, the following are the entry
**      conditions:
**        D(A)=Starting address of buffer,
**        A(A)=Length of buffer(in bytes).
**
**      The handler may use any CPU registers
**        except D0, R0 and the status bits.
**
**      The handler has 3 stack levels (RSTK) available.
**
** History:
**
**    Date      Programmer            Modification
**  --------   ----------   ----------------------------------
**  06/25/82   B.S.         Updated documentation
```

```
144                **
145                **********************************************************************
146                **********************************************************************
147                **********************************************************************
148                **********************************************************************
149                **
150                ** Name:    xPART2  -  POLL handler for part 2 of output
151                **
152                ** Category:   POLL
153                **
154                ** Purpose:
155                **      Send character(s) to the current output device
156                **
157                ** Entry:
158                **      P=0, HEXMODE
159                **      D(A)=Start address of buffer
160                **      A(A)=Length of bufer (in bytes)
161                **
162                ** Exit:
163                **      P=0, HEXMODE
164                **      D1 points past last character sent (next output char)
165                **
166                ** Uses.......
167                **  Inclusive: A, B, C, D, D1, FUNCD0, FUNCD1, FUNCR0, FUNCR1
168                **             Does NOT alter D0, Status bits
169                **
170                ** Stk lvls:   <4
171                **
172                ** NOTE:
173                **      DO NOT ALTER D0 OR STATUS BITS!!!!
174                **
175                ** History:
176                **
177                **    Date     Programmer              Modification
178                **  --------   ----------   ----------------------------------
179                **  11/17/82   N.Z.         Added documentation
180                **  01/27/83   M.B.         Documented exit conditions
181                **
182                **********************************************************************
183                **********************************************************************
184 17DE3 137   =SENDIT CD1EX
185 17DE6 1F00          D1=(5) =SAVSTK        Point to buffer pointer
          000
186 17DED AF0           A=0    W              Zero out upper nibbles
187 17DF0 143           A=DAT1 A              Read in buf start addr
188
189                *                      C=Start addr A=End addr
190 17DF3 D7     SEND10 D=C    A              Copy start addr to D
191 17DF5 EA            A=A-C  A              Calc len of buffer in nibs
192 17DF7 81C           ASRB                  Divide by 2 (Len in bytes)
193                *
194 17DFA 7303  =SEND20 GOSUB  D1@POS         Point to position
195 17DFE 14F           C=DAT1 B              Read in position
196 17E01 A62           C=C+A  B              Add buf len to position
197 17E04 14D   SEND30 DAT1=C B              Write out new position
```

```
198 17E07 1F00        D1=(5) (=STMTR0)+oHNDLR  Point to I/O handler addr
          000
199 17E0E 147         C=DAT1 A                 Read in handler addr
200 17E11 06          RSTK=C                   Push it on stack
201 17E13 03          RTNCC                    Call I/O handler
```

```
202                 STITLE SENDWD - Send width sized chunks
203               ****************************************************************
204               ****************************************************************
205               **
206               ** Name:(S) SENDWD  -  Send Out Width-Sized Chunks to Device
207               ** Name:(S) SNDWD+  -  Send Out Width-Sized Chunks to Device
208               **
209               ** Category:   DSPUTL
210               **
211               ** Purpose:
212               **      Send out width-sized chunks to display/printer device.
213               **
214               ** Entry:
215               **      STMTRO must have been set up correctly by CKINFO
216               **      Status bit InhEOL (4):
217               **          0= send out initial CR-LF if buffer won't fit in
218               **             first width-sized chunk
219               **             (only if position .ne. 0)
220               **          1= start sending out buffer immediately, regardless
221               **             if the buffer won't fit on the first line.
222               **      SENDWD:
223               **        A(A)= #characters (#bytes) in output buffer.
224               **        D1 points to output buffer.
225               **      SENDW+:
226               **        B(A)= #character (#bytes) in output buffer.
227               **        C(A) points to output buffer.
228               **
229               ** Exit:
230               **      P      = 0
231               **      Carry set
232               **      A(A)   = 0
233               **
234               ** Calls:      CSLWP9, CSRWP9, SENDEL, SEND20, D1@POS, B2C95,
235               **             CSLWP, CSRWP
236               **
237               ** Uses.......
238               **  Exclusive: A(A), B(A), C(W), D(A), P, D1,     R2
239               **  Inclusive: A(W), B(W), C(W), D(W), P, D1, R1, R2
240               **             Does not change D0, Status
241               **
242               ** Stk lvls:   4
243               **
244               ** NOTE:
245               **      DO NOT CHANGE D0 OR STATUS BITS!!!
246               **
247               ** Detail:
248               **              15 14 13 12 11 10 9  8  7  6  5  4  3  2  1  0
249               **   R2 usage:    |#chars in bfr |  entry D1    |              |
250               **
251               ** History:
252               **
253               **    Date     Programmer             Modification
254               **  --------   ----------   ---------------------------------
255               **  08/26/82   M.B.         Wrote routine.
256               **
```

```
257                ****************************************************************
258                ****************************************************************
259 17E15 8A8  =SENDWD ?A=0   A              Any more chars?
260 17E18 00           RTNYES                No.
261 17E1A D8           B=A    A              Counter to B.
262 17E1C 137          CD1EX                 Save D1= buffer pointer
263 17E1F D7   =SNDWD+ D=C    A                in D.
264 17E21 7CD2         GOSUB  D1@POS         D1 points to print posn.
265 17E25 171          D1=D1+ 2              To width.
266 17E28 D0           A=0    A
267 17E2A 14B          A=DAT1 B              A(A)= disp/print width.
268 17E2D 968          ?A=0   B              Zero width?
269 17E30 35           GOYES  SNDW19         Yes. Send out entire buffer.
270 17E32 9EA          ?A<=C  B              Position already past width?
271 17E35 71           GOYES  SNDW09         Yes. Send EOL.
272 17E37 EA           A=A-C  A              W-P = #chars to fill width.
273 17E39 8B8          ?A>=B  A              Enough chars to fill width?
274 17E3C 74           GOYES  SNDW19         Yes. Exhaust chars in buffer.
275 17E3E E8           B=B-A  A              Too many.
276                                          #chars in bfr after send.
277 17E40 874          ?ST=1  InhEOL         Chars left after send. Fresh EOL?
278 17E43 93           GOYES  SNDW17         No.
279 17E45 96A          ?C=0   B              At position zero already?
280 17E48 43           GOYES  SNDW17         Yes. No extra EOL.
281 17E4A C8           B=B+A  A              Restore B=total #chars in buffer.
282                *
283 17E4C 7640  SNDW09 GOSUB  B2C95          C(9-5)=#chrs remaining in buffer.
284 17E50 DB           C=D    A              D1 to C(A).
285 17E52 2E           P=     14             #chars to C(14-10),
286 17E54 8F00         GOSBVL =CSLWP           D1 to C(9-5).
            000
287 17E5B 10A          R2=C                  Store all in R2.
288 17E5E 7F5F         GOSUB  SENDEL         Send EOL.
289 17E62 11A          C=R2                  Restore #chars, D1.
290 17E65 2E           P=     14
291 17E67 8F00         GOSBVL =CSRWP
            000
292 17E6E 135          D1=C                  D1 restored, points to buffer.
293 17E71 8E00  SNDW11 GOSUBL =CSRW9j
            00
294 17E77 DA           A=C    A              #chars remaining in buffer to A.
295 17E79 5B9          GONC   SENDWD         (BET) Send out next chunk.
296                *
297                *
298 17E7C 7610  SNDW17 GOSUB  B2C95          C(9-5)=#chars remaining in bfr.
299 17E80 570          GONC   SNDW21         (BET) Send out width-sized chunk.
300                *
301 17E83       SNDW19                       Send out all remaining chars.
302 17E83 D4           A=B    A              A(A)= #chars left in buffer.
303 17E85 AF2          C=0    W              "No chars left".
304 17E88 10A  SNDW21 R2=C                   Store #chars left in R2(9-5).
305 17E8B 7B6F         GOSUB  SEND20         Send to device.
306 17E8F 11A          C=R2                  #chars left to C(9-5)
307 17E92 6EDF         GOTO   SNDW11         To C(A) and send out remaining.
308                *
```

```
    309 17E96 D9    B2C95  C=B    A              Store B(A)
    310 17E98 8C00  CSLWP9 GOLONG =CSLW9j          in C(9-5).
              00
```

```
311                     STITLE DPART2 - DISP IO handler
312                 ****************************************************************
313                 ****************************************************************
314                 **
315                 ** Name:(S) DPART2  -  IO Handler For Built-In Display
316                 **
317                 ** Category:   EXCUTL
318                 **
319                 ** Purpose:
320                 **      Sends output to display devices at execution time
321                 **
322                 ** Entry:
323                 **      P      = 0
324                 **      D(A)=Start address of buffer
325                 **      A(A)=Length of bufer (in bytes)
326                 **
327                 ** Exit:
328                 **      P      = 0
329                 **      D1 points past last char sent (to next output char)
330                 **
331                 ** Calls:      CSRWP9,DSPCHA,CK"ON"
332                 **
333                 ** Uses.......
334                 **  Exclusive: R1(W), R2(A), A(W),       C(W),       D1
335                 **  Inclusive: R1(W), R2(A), A(W), B(W), C(W), D(W), D1
336                 **
337                 ** Stk lvls:   3
338                 **
339                 ** Detail:
340                 **                 15 14 13 12 11 10 9  8  7  6  5  4  3  2  1  0
341                 **   R1 usage:                      |  entry D0    | buffer D1   |
342                 **
343                 **   R2 usage:    |              |              | counter #chr|
344                 **
345                 ** History:
346                 **
347                 **    Date     Programmer             Modification
348                 **   --------   ----------   ---------------------------------
349                 **   10/19/82   B.S.         Updated documentation
350                 **   01/27/83   M.B.         Documented exit conditons
351                 **
352                 ****************************************************************
353                 ****************************************************************
354 17E9E A500         REL(5) =DPART3        Address of part 3 of I/O Handler
        0
355                 *
356 17EA3 D8    =DPART2 B=A    A              Preserve upper nibbles
357 17EA5 112          A=R2                    of R2.
358 17EA8 D4           A=B    A
359 17EAA 136          CDOEX                  Save D0.
360 17EAD 77EF         GOSUB  CSLWP9          D0 to C(9-5).
361 17EB1 DB           C=D    A               Output buffer address.
362 17EB3 135          D1=C                   D1 points to output buffer.
363 17EB6 CC     DPRT21 A=A-1  A              Count #buffer chars.
364 17EB8 443          GOC    DPRT25
```

```
365 17EBB 102          R2=A                Save counter in R2(A).
366 17EBE 14B          A=DAT1 B            Read character.
367 17EC1 171          D1=D1+ 2            To next char.
368 17EC4 137          CD1EX               Save D1 in C.
369 17EC7 86C          ?ST=0  =Except      Has attention key been pressed?
370 17ECA C0           GOYES  DPRT23       No. Display character.
371 17ECC 8F00         GOSBVL =CK"ON"      Maybe. Check "ON" key.
          000
372 17ED3 591          GONC   DPRT25       NC= Attn key hit.
373 17ED6 109  DPRT23  R1=C                D1 to R1.
374 17ED9 8F00         GOSBVL =DSPCHA      Output character.
          000
375 17EE0 119          C=R1                Restore D1
376 17EE3 135          D1=C                  to point to output buffer.
377 17EE6 112          A=R2                Counter to A.
378 17EE9 6CCF         GOTO   DPRT21       Next character.
379              *
380 17EED 8E00  DPRT25 GOSUBL =CSRW9j      Restore D0 from R1(9-5).
          00
381 17EF3 134          D0=C
382 17EF6 03           RTNCC
383              *
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) DPART3  -  Finish up DISP line
                **
                ** Category:   EXCUTL
                **
                ** Purpose:
                **      Puts finishing touches on a DISP statement line,
                **      specifically, causing the display to be built and
                **      the line to be scrolled if necessary.
                **
                ** Entry:
                **      P       =  0
                **      InhEOL(ST4) set if CR/LF has not just been sent to
                **      display
                **
                ** Exit:
                **      P       =  0
                **
                ** Calls:       DOSCRL
                **
                ** Uses.......
                **  Inclusive: A,B,C,D,D0,D1
                **
                ** Stk lvls:   5
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ------------------------------------
                **  11/01/83    B.S.         Added documentation
                **
                ********************************************************************
                ********************************************************************
17EF8           =DPART3
17EF8 864               ?ST=0  InhEOL          Display already built by CR/LF?
17EFB 00                RTNYES                 Yes, then just return
17EFD 8D00              GOVLNG =DOSCRL         Scroll display
      000
```

```
                        STITLE PRINT Execution
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:    PRINT   -  PRINT statement execution
                    **
                    ** Category:   STEXEC
                    **
                    ** Purpose:
                    **      Implements PRINT statement
                    **
                    ** Entry:
                    **      DO points past statement token
                    **
                    ** Exit:
                    **      Exits through NXTSTM
                    **
                    ** Algorithm:
                    **      Sets STMTRO(0) to PRINT statement type
                    **      Jumps to PRINT*
                    **
                    ** History:
                    **
                    **    Date     Programmer              Modification
                    **  --------   ----------   ----------------------------------
                    **  10/18/83   B.S.         Updated Documentation
                    **
                    ****************************************************************
                    ****************************************************************
453 17F04 0000          REL(5) =PRNTDC
          0
454 17F09 0000          REL(5) =PRTP
          0
455 17F0E 14A  =PRINT   A=DAT0 B
456 17F11 20            P=     0
457 17F13 3132          LCASC  \#\
458 17F17 962           ?A=C   B
459 17F1A 90            GOYES  PRINT.
460 17F1C 31F1          LC(2)  (PRINTt)*16+#F
461 17F20 561           GONC   PRINT*          B.E.T.
462 17F23 8C00  PRINT.  GOLONG =PRINT#
          00
```

```
                        STITLE DISP statement execution
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:    DISP    -  DISP statement execute
                    **
                    ** Category:   STEXEC
                    **
                    ** Purpose:
                    **       Implements DISP statement execute
                    **
                    ** Entry:
                    **       DO points past statement token
                    **
                    ** Exit:
                    **       Exits through NXTSTM
                    **
                    ** Algorithm:
                    **       Sets STMTRO(0) to DISP statement type
                    **       Falls to PRINT*
                    **
                    ** History:
                    **
                    **    Date     Programmer              Modification
                    **  --------   ----------   ----------------------------------
                    **  10/18/83   B.S.         Added Documentation
                    **
                    ****************************************************************
                    ****************************************************************
17F29 00000         REL(5) =DISPDC
      0
17F2E 00000         REL(5) =DISPP
      0
17F33 31F0 =DISP    LC(2)  (DISPt)*16+#F  Code for DISP statement
              *     Now fall through to PRINT*
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) PRINT*  -  PRINT class statement execution
                **
                ** Category:   STEXEC
                **
                ** Purpose:
                **      Implements PRINT class statement execution.  This
                **      includes DISP and PRINT.
                **
                ** Entry:
                **      P      =  0
                **      C(0) = PRINT class statement class number
                **             0 --> DISP
                **             1 --> PRINT
                **             2 --> OUTPUT
                **             3 --> PLOT
                **
                ** Exit:
                **      Exits through NXTSTM
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------   ------------------------------------
                **  11/01/83   B.S.         Added documentation
                **
                ********************************************************************
                ********************************************************************
17F37 7DF5      =PRINT* GOSUB  CKINF+          Initialize Statement Scratch
17F3B 14A               A=DAT0 B               Read next byte
17F3E 3100              LC(2)  =tUSING
17F42 966               ?A#C   B               Is it a USING token?
17F45 F0                GOYES  DISP05          No, then continue
17F47 8C00              GOLONG =USING          Yes, then jump to USING
      00
                *-
                *-
17F4D 8D00      =collap GOVLNG =COLLAP
      000
                *-
                *-
17F54 844        DISP05 ST=0   InhEOL          Allow CR/LF
17F57 72FF       DISP10 GOSUB  =collap         Collapse expression stack
17F5B 8F00              GOSBVL =FINDDO
      000
17F62 00                CON(2) =tSEMIC         Semicolon
17F64 310               REL(3) DISP15
17F67 00                CON(2) =tCOMMA         Comma
17F69 D10               REL(3) DISP30
17F6C 00                CON(2) =tTAB           TAB
17F6E 080               REL(3) TABOO
17F71 00                CON(2) 0               Otherwise
```

```
548 17F73 6611          GOTO   DISP50
549             *-
550             *-
551 17F77 854   DISP15  ST=1   InhEOL          Inhibit CR/LF
552 17F7A 161   DISP20  D0=D0+ 2               Move program counter
553 17F7D 59D           GONC   DISP10          (B.E.T.)
554             *-
555             *-
556 17F80 8C00  =MEMERj GOLONG =nomem          GOTO  MEMERR
          00
557             *-
558             *-
559 17F86       DISP30
560 17F86 7771          GOSUB  D1@POS
561 17F8A DA            A=C    A               Copy position to A
562 17F8C 3151          LC(2)  21
563 17F90 D5            B=C    A
564 17F92 8B4   CMMA10  ?B>A   A               Past position yet?
565 17F95 70            GOYES  CMMA20          Yes, then exit loop
566 17F97 C1            B=B+C  A               No, then add another 21
567 17F99 58F           GONC   CMMA10          (B.E.T.)
568             *-
569             *-
570             *  B now points at beginning of next print field
571 17F9C 171   CMMA20  D1=D1+ 2
572 17F9F 14F           C=DAT1 B               Read width
573 17FA2 CE            C=C-1  A               Width-1
574 17FA4 470           GOC    CMMA30          Skip if width=inf
575 17FA7 9E1           ?B>C   B               Is print-field-start > width?
576 17FAA D1            GOYES  CMMA60          Yes, then send Endline
577 17FAC E0    CMMA30  A=A-B  A               Position - print-field-start =
578                                             negative number of blanks needed
579 17FAE 7B96          GOSUB  GETAVM          D1=AVE,D=AVMS
580 17FB2 3102          LCASC  \ \             ASCII blank
581 17FB6 7A45  CMMA50  GOSUB  STKCHR
582 17FBA E4            A=A+1  A               Increment neg count
583 17FBC 59F           GONC   CMMA50          Loop back until done
584 17FBF 702E          GOSUB  SENDIT          Send blanks to device
585 17FC3 63BF          GOTO   DISP15          Finish up
586             *-
587             *-
588 17FC7 76FD  CMMA60  GOSUB  SENDEL          Send EOL.
589 17FCB 6BAF          GOTO   DISP15          Next item.
590             *-
591             *-
592 17FCF 136   TABWRN  CDOEX
593 17FD2 10B           R3=C                   Save PC
594 17FD5 3300          LC(4)  =eIVTAB
          00
595 17FDB 8F00          GOSBVL =MFWRQ8         Give warning(Beep,errl,delay)
          000
596 17FE2 11B           C=R3
597 17FE5 134           D0=C                   Restore PC to D0
598 17FE8 D0            A=0    A
599 17FEA 6420          GOTO   TAB10
```

```
600                *-
601                *-
602 17FEE 161    TAB00  D0=D0+ 2            Move pointer past TAB token
603 17FF1 7335           GOSUB  EXPEXj       Evaluate expression
604 17FF5 8F00           GOSBVL =POP1N       Pop tab value from stack
          000
605 17FFC 42D            GOC    TABWRN       Give warning if complex
606 17FFF 94C            ?A#0   S           Is it >=0?  (Temporary....)
607 18002 DC             GOYES  TABWRN       Yes, then give warning
608 18004 8E00           GOSUBL =FLTDH       Convert to hex
          00
609 1800A CC             A=A-1  A           Convert to option base zero
610 1800C 42C            GOC    TABWRN       Give warning if it was zero
611 1800F 100    TAB10   R0=A               Save tab value
612 18012 7C25           GOSUB  CKINFO
613 18016 77E0           GOSUB  D1@POS       Point at display position
614 1801A 110            A=R0               Recall TAB position
615 1801D D5             B=C    A
616 1801F 171            D1=D1+ 2           Point at width
617 18022 14F            C=DAT1 B           Read width
618 18025 96E            ?C#0   B           Is width infinity?
619 18028 50             GOYES  TAB20       No, then skip
620 1802A B26            C=C+1  XS          Use width=256
621 1802D 8B2    TAB20   ?A<C   A           Is TAB>Width?
622 18030 70             GOYES  TAB30       No, then  continue
623 18032 EA             A=A-C  A           Do mod Width reduction
624 18034 58F            GONC   TAB20       (B.E.T.)
625                *-
626                *-
627 18037 E0     TAB30   A=A-B  A           Subtract current position
628 18039 100            R0=A
629 1803C 5B0            GONC   TAB40       Skip if it will fit on curr line
630 1803F C0             A=A+B  A
631 18041 100            R0=A
632 18044 797D           GOSUB  SENDEL       Yes, then send EndLine first
633 18048 8F00   TAB40   GOSBVL =BLANKC      C=8 spaces
          000
634 1804F 1F00           D1=(5) =FUNCR0
          000
635 18056 1557           DAT1=C W
636 1805A 3400   TAB50   LC(5)  =FUNCR0
          000
637 18061 D7             D=C    A           D=Start address
638 18063 D2             C=0    A
639 18065 308            LC(1)  8
640 18068 110            A=R0
641 1806B EA             A=A-C  A           More than 8 spaces left?
642 1806D 4F0            GOC    TAB60       No, then send last spaces
643 18070 100            R0=A               # of remaining spaces
644 18073 DA             A=C    A           A(A)=8
645 18075 718D           GOSUB  SEND20
646 18079 60EF           GOTO   TAB50
647                *-
648                *-
649 1807D CA     TAB60   A=A+C  A           Undo subtraction
```

```
650 1807F 777D          GOSUB  SEND20          Send last spaces
651 18083 844   DISP1j  ST=0   InhEOL          Allow CR/LF
652 18086 60DE          GOTO   DISP10          Temporary do nothing ****
653             *-
654             *-
655 1808A 310F  DISP50  LCHEX  F0
656 1808E 9E2           ?A<C   B               Is it @, eol, else, ! token?
657 18091 92            GOYES  DISP80          No, then it must be an expr
658 18093 752D          GOSUB  CRLFCK
659             ********************************************************************
660             ********************************************************************
661             **
662             ** Name:(S) PART3   -  Finishes up a PRINT class statement
663             **
664             ** Category:   STEXEC
665             **
666             ** Purpose:
667             **      This is the 3rd part of PRINT class statements.  It
668             **      calls the appropriate routine to finish up the current
669             **      line.
670             **
671             ** Entry:
672             **      P      =  0
673             **      STMTRO set up by CKINFO
674             **
675             ** Exit:
676             **      Exits through NXTSTM
677             **
678             ** Calls:      xPART3
679             **
680             ** History:
681             **
682             **    Date     Programmer             Modification
683             **  --------   ----------   ---------------------------------
684             **  11/09/83   B.S.         Added documentation
685             **
686             ********************************************************************
687             ********************************************************************
688 18097 3400 =PART3  LC(5)  =NXTSTM
          000
689 1809E 06           RSTK=C                 Push address of NXTSTM on stack
690 180A0 1F00          D1=(5) (=STMTRO)+oHNDLR
          000
691 180A7 7DA5          GOSUB  D1=@D1          D1=handler address
692 180AB 1C4           D1=D1- 5               Move back 4 nibbles
693 180AE 147           C=DAT1 A               Read in end handler offset
694 180B1 133           AD1EX
695 180B4 C2            C=C+A  A               Add offset to base to get handler
696                                              address
697 180B6 06            RSTK=C                 Push end handler address on stack
698 180B8 03            RTNCC                  Jump to end handler,
699                                              then rtn to NXTSTM
700             *-
701             *-
702 180BA 7A64  DISP80  GOSUB  EXPEXj          Evaluate expression
```

```
    703 180BE 7084          GOSUB  CKINFO        Verify info okay
    704 180C2 14B           A=DAT1 B             Re-read stack header
    705 180C5 B04           A=A+1  P
    706 180C8 A64           A=A+A  B             Check for F0 or F8
    707 180CB 968           ?A=0   B             Is it a string?
    708 180CE 31            GOYES  DISP90        Yes, then okay
    709               *
    710 180D0 136           CDOEX                Save D0 in C
    711 180D3 7E30          GOSUB  PUTRES        Put expr in result register.
    712 180D7 134           D0=C                 Restore D0
    713 180DA 851           ST=1   Blanks        Want leading and trailing blanks
    714 180DD 7B60          GOSUB  STR$RT        No, then convert it to one
    715 180E1 8E00  DISP90  GOSUBL =REV$         Reverse chars in string
              00
    716 180E7 844           ST=0   InhEOL        Allow CR/LF
    717 180EA 171           D1=D1+ 2             Skip F0 (string head)
    718               *
    719 180ED AF0           A=0    W
    720 180F0 143           A=DAT1 A             #nibbles in string expr.
    721 180F3 81C           ASRB                 #chars in string.
    722 180F6 17D           D1=D1+ 14            Skip to text.
    723 180F9 781D          GOSUB  SENDWD        Send out in width-sized chunks.
    724 180FD 695E          GOTO   DISP10        Continue processing statement
    725               *-
    726               *-
    727               *-
    728 18101 1F00  D1@POS  D1=(5) (=STMTRO)+oPOSIT
              000
    729 18108 147           C=DAT1 A
    730 1810B 135           D1=C
    731 1810E D2            C=0    A
    732 18110 14F           C=DAT1 B
    733 18113 03            RTNCC
    734               *-
```

```
735                     STITLE PUTRES - Store into RES register
736                 ******************************************************************
737                 ******************************************************************
738                 **
739                 ** Name:(S) PUTRES  -  Put Numeric Result Into RES
740                 **
741                 ** Category:   EXCUTL
742                 **
743                 ** Purpose: Put numeric expression in RES register.
744                 **
745                 ** Entry:    D1 points to start of numeric expression on stack
746                 **            (or any desired location).
747                 **
748                 ** Exit:     Carry clear: real.  Carry set: complex.
749                 **           D1= same value as entry.
750                 **           P=0.  Sets HEX mode.
751                 **
752                 **
753                 ** Calls:    POP1N
754                 **
755                 ** Uses:     P, A(W), B(0), D0
756                 **           R0 if complex.
757                 **
758                 ** Stk lvls:   1
759                 **
760                 ** Algorithm:
761                 **      Call POP1N (express purpose of checking numeric arg)
762                 **      Set D0= RESREG
763                 **      If complex, read 34 nibbles from the Math stack to
764                 **        put in the RES register.
765                 **      If real, simply write A(W) into the RES register.
766                 **      Returns D1 to original value.
767                 **
768                 ** History:
769                 **
770                 **    Date      Programmer              Modification
771                 **  --------   ----------   ----------------------------------
772                 **  08/26/82   M.B.         Wrote routine
773                 **
774                 ******************************************************************
775                 ******************************************************************
776 18115 8F00 =PUTRES GOSBVL =POP1N          Check argument for numeric.
           000
777 1811C 04           SETHEX                POP1N may have left DEC mode.
778 1811E 1B00         D0=(5) =RESREG        Point to RES register.
           000
779 18125 5D1          GONC   PUTRE9         NC= real.
780 18128 16F          D0=D0+ 16             Complex. Move D0 to middle
781 1812B 161          D0=D0+ 2                of RES register.
782 1812E 2E           P=     14             Counter.
783                 *
784 18130 1507  PUTRE3 DAT0=A W              Put real part into RES.
785 18134 1C8          D1=D1- 9              Move partially into
786 18137 188          D0=D0- 9                imaginary part.
787 1813A 1537         A=DAT1 W
```

```
    788 1813E 0C          P=P+1                 Count twice.
    789 18140 5FE         GONC   PUTRE3
    790           *                             Carry set on loop exit.
    791 18143 1507  PUTRE9 DATO=A W
    792 18147 01          RTN                   (Preserve carry!)
    793           *                             Carry Set: complex. Clear: real.
```

```
794                 $TITLE STR$SB - Convert number to string
795             ****************************************************************
796             ****************************************************************
797             **
798             ** Name:(S) STR$SB  -  Convert Number to String
799             ** Name:(S) STR$00  -  Convert Number to String(Generic)
800             **
801             ** Category:   CONVRT
802             **
803             ** Purpose:
804             **      Pops a number off stack and pushes a string on stack
805             **      containing ASCII representation in current display
806             **      setting.
807             **      STR$SB is a subroutine which returns a string without
808             **      leading and trailing blanks surrounding the number.
809             **      STR$00 is a generic routine which will either return
810             **      when done or jump to EXPR.  It may or may not output
811             **      leading and trailing blanks.
812             **
813             ** Entry:
814             **      P      = 0
815             **      D1 points to top of stack
816             **
817             **      STR$00:
818             **        Return (S0) set iff return is desired
819             **                otherwise jumps to EXPR when done.
820             **        Blanks (S1) set iff leading and trailing blanks
821             **             are desired.
822             **
823             ** Exit:
824             **      P      = 0
825             **      D1 points to string
826             **      Exits to MEMERR if memory overflows
827             **
828             ** Calls:      POP1N,FMTNUM,STKCHR,NAN?,FMTPRP,ADHEAD,D=AVMS,
829             **             DSFORM
830             **
831             ** Uses.......
832             **  Exclusive: D1,S0,S1,C(A),D(A)
833             **  Inclusive: A,B,C,D(A),S0,R0,R1,R2
834             **
835             ** Stk lvls:   2
836             **
837             ** Detail:
838             **      Pops an numeric item off expression stack and
839             **      checks the current display format.
840             **        Standard format: If the number can be represented
841             **            without losing accuracy in 12 digits plus
842             **            optionally a decimal point it will be, else
843             **            scientific notation will be used and all
844             **            significant digits will be shown.
845             **
846             **        FIX n: Display n places past the decimal point
847             **            with rounding.  If result is longer than
848             **            13 digits, defaults to SCI n.
```

```
849                **
850                **       SCI n: Display n+1 significant digits in
851                **           scientific notation with rounding.
852                **           (1. <= mantissa <= 9.999999...)
853                **
854                **       ENG n: Display n+1 significant digits in
855                **           engineering notation with rounding.
856                **           (1. <= mantissa <= 999.9999...;
857                **            exponent divisible by 3)
858                **
859                ** History:
860                **
861                **    Date     Programmer          Modification
862                **  --------   ----------  ----------------------------------
863                **  07/20/82   B.S.        Updated documentation
864                **
865                ********************************************************************
866                ********************************************************************
867 18149 841    =STR$SB ST=0    Blanks          Lead/trailing blanks suppressed
868 1814C 850     STR$RT ST=1    Return          Return to caller
869 1814F 6C00           GOTO    STR$00
870                *-
871                *-
872 18153 8              NIBHEX 8               Argument must be numeric
873 18154 11             NIBHEX 11              Argument count range
874 18156 840  =STR$     ST=0    Return          Jump to EXPR when done
875 18159 841            ST=0    Blanks          Not display form
876 1815C 7335 =STR$00   GOSUB   d=avms          D(A)=(AVMEMS)
877 18160 8F00           GOSBVL  =POP1N          Check for number atop stack
          000
878 18167 04             SETHEX                  Done with POP1N
879 18169 4E1            GOC     STR$CP          Handle complex number
880 1816C 7170           GOSUB   FMTPRP          Prepare for formatting
881 18170 7B70           GOSUB   NAN?            Is it a NaN?
882 18174 470            GOC     STR$.5          Yes, then don't check sign
883 18177 94C            ?A#0    S               Is number negative?
884 1817A 60             GOYES   STR$01          No, don't put out leading space
885 1817C 7180  STR$.5   GOSUB   DSFORM          Output leading blank if required
886 18180 7A80  STR$01   GOSUB   FMTNUM          Format number
887 18184 6E20           GOTO    STR$30          Finish up
888                *-
889                *-
890 18188 7550  STR$CP   GOSUB   FMTPRP          Prepare for formatting
891 1818C 7170           GOSUB   DSFORM          Output leading blank if required
892 18190 3182           LCASC   \(\
893 18194 7C63           GOSUB   STKCHR          Output left parenthesis
894 18198 7270           GOSUB   FMTNUM          Format real part
895 1819C 31C2           LCASC   \,\
896 181A0 7063           GOSUB   STKCHR          Output comma
897 181A4 110            A=R0                    Recall imaginary part
898 181A7 7360           GOSUB   FMTNUM          Format imaginary part
899 181AB 3192           LCASC   \)\
900 181AF 7153           GOSUB   STKCHR          Output right parenthesis
901 181B3 7A40  STR$30   GOSUB   DSFORM          Output trailing blank if required
902                * Now fall into ADHEAD
```

```
903                     STITLE ADHEAD - Add string header to stack
904                *****************************************************************
905                *****************************************************************
906                **
907                ** Name:(S) ADHEAD  -  Add String Header
908                **
909                ** Category:   MTHSTK
910                **
911                ** Purpose:
912                **      Adds string header to string on stack
913                **
914                ** Entry:
915                **      R1(A)=Start of stack item(hi mem)
916                **      D1=End of stack item(low mem)
917                **      S0 set iff RTN desired (jumps to EXPR otherwise
918                **      D(A)=(AVMEMS)
919                **      P=0
920                **
921                ** Exit:
922                **      D1 points at string header on stack
923                **
924                ** Calls:      STKCH+
925                **
926                ** Uses.......
927                **  Exclusive: A(A),C(W),D1
928                **  Inclusive: A(A),C(W),D1
929                **
930                ** Stk lvls:   0
931                **
932                ** Detail:
933                **      R1 should have been used to store stack pointer
934                **      before putting string on stack.  As the string
935                **      was added to stack, D1 should have been decremented
936                **      to keep it pointed at the last char of string.
937                **      This routine can then be used to tack on the string
938                **      header (F0111110000000000) where 11111 is the length
939                **      of the string.
940                **
941                ** History:
942                **
943                **    Date     Programmer              Modification
944                **  --------   ----------   ----------------------------------
945                **  07/20/82   B.S.         Updated documentation
946                **
947                *****************************************************************
948                *****************************************************************
949 181B7 119   =ADHEAD C=R1
950 181BA 133           AD1EX
951 181BD 131           D1=A                    Copy D1 to A
952 181C0 E2            C=C-A  A                Calculate string length
953 181C2 BF2           CSL    W
954 181C5 BF2           CSL    W                Make room for string identifier
955 181C8 30F           LCHEX  F                String identifier
956 181CB 1CF    STR$40 D1=D1- 16
957 181CE 7533          GOSUB  STKCH+           Check if enough room
```

```
    958 181D2 15D6          DAT1=C 7              Write out string header
    959 181D6 870           ?ST=1  Return         Is a return requested?
    960 181D9 00            RTNYES                Yes, then return
    961 181DB 8C00  EXPRj   GOLONG =Expr          No, then go to EXPR
              00
    962               *-
    963               *-
    964 181E1 17F    FMTPRP D1=D1+ 16
    965 181E4 137           CD1EX
    966 181E7 135           D1=C
    967 181EA 109           R1=C                  Save begin of string
    968 181ED 01            RTN
    969               *-
    970               *-
    971 181EF 04     NAN?   SETHEX
    972 181F1 B24           A=A+1  XS
    973 181F4 A2C           A=A-1  XS
    974 181F7 500           RTNNC                 Return with carry clear
    975                                             if not NAN or INF
    976 181FA 96C           ?A#0   B              Is it a NAN?
    977 181FD 00            RTNYES                Yes, then return with carry set
    978 181FF 03            RTNCC                 No, then return with carry clear
    979               *-
    980               *-
    981 18201 861    DSFORM ?ST=0  Blanks         Blank required?
    982 18204 00            RTNYES                No, then return
    983 18206 3102          LCASC  \ \            Yes, then add one
    984 1820A 69F2          GOTO   STKCHR
    985               *-
    986               *-
```

```
987                     STITLE CLCSTR - Turn Number into ASCII
988             ****************************************************************
989             ****************************************************************
990             **
991             ** Name:    CLCSTR  -  Turn Number into ASCII
992             **
993             ** Category:   DSPUTL
994             **
995             ** Purpose:
996             **      Turn a 12-digit floating-point number into its ASCII
997             **      representation for display purposes and place on
998             **      stack.
999             **
1000            ** Entry:
1001            **      A=Number to be displayed.
1002            **      D1=stack pointer.
1003            **      R1[A] points to top-of-stack before we started writing
1004            **        to it (may include leading space before CLCSTR's
1005            **        output).
1006            **      D[A]=(AVMEMS).
1007            **
1008            ** Exit:
1009            **      P=0.
1010            **      D1 = New stack pointer
1011            **
1012            ** Calls:      STKCHR,NAN?
1013            **
1014            ** Uses.......
1015            **              A,B,C,D[15-13],D1,P
1016            **
1017            ** Stk lvls:   1
1018            **
1019            ** Algorithm:
1020            **      A -> R2.
1021            **      If A=NAN, write "NaN" to stack and RTN.
1022            **      If A negative, write "-".
1023            **      If A=INF, write "Inf" and RTN.
1024            **      Read display flags; clear upper 2 bits of DSPFMT nib.
1025            **      If Fix, Sci, or Eng, goto FXSCEN.
1026            **      Dope up phony fix or sci display based on the
1027            **        following:
1028            **          Define Q=#sig digits - 1.
1029            **          If exp>0 then
1030            **            if exp>11, display as SCI Q,
1031            **            else if Q-exp >= 0 display as FIX (Q-exp),
1032            **                 else display as FIX 0.
1033            **          Else if Q-exp > 12 display as SCI Q.
1034            **               else display as FIX (Q-exp).
1035            **      If displaying as FIX (something), goto FXSCEN with
1036            **        upper bit set in dspfmt (dspfmt = 9),
1037            **      Else goto FXSCEN with dspfmt = 2.
1038            **
1039            ** History:
1040            **
1041            **    Date     Programmer              Modification
```

```
1042            **  --------  ----------  ---------------------------------
1043            **  10/28/82  NM          Rewrote.
1044            **
1045            ********************************************************************
1046            ********************************************************************
1047 1820E     =CLCSTR
1048 1820E 04   FMTNUM SETHEX
1049 18210 102         R2=A                    Save number being formatted in R2
1050 18213 78DF        GOSUB  NAN?             Is it a NAN?
1051 18217 AF2         C=0    W
1052 1821A 5B1         GONC   STR$05           No, then continue
1053 1821D 35E4        LCASC  \NaN\            Output "NaN"
           16E4
1054 18225 7BD2 STKLST GOSUB  STKCHR
1055 18229 BF6         CSR    W
1056 1822C BF6         CSR    W
1057 1822F 8AE         ?C#0   A
1058 18232 3F          GOYES  STKLST
1059 18234 01          RTN
1060            *-
1061            *-
1062 18236 948  STR$05 ?A=0    S               Is sign positive?
1063 18239 A0          GOYES  STR$10           Yes, then skip
1064 1823B 31D2        LCASC  \-\
1065 1823F 71C2        GOSUB  STKCHR           Output minus sign
1066 18243 B24  STR$10 A=A+1  XS
1067 18246 A2C         A=A-1  XS               Set carry if INF
1068 18249 3594        LCASC  \fnI\
           E666
1069 18251 43D         GOC    STKLST           If INF, output "Inf" to stack
1070            *-
1071            *-
1072 18254 133         AD1EX
1073 18257 1F00        D1=(5) =DSPFMT
           000
1074 1825E 14F         C=DAT1 B                Read display format & dgt
1075 18261 0B          CSTEX
1076 18263 842         ST=0   2                Clear upper 2 bits of dspfmt
1077 18266 843         ST=0   3
1078 18269 0B          CSTEX
1079 1826B 133         AD1EX
1080 1826E 112         A=R2                    Fetch number
1081 18271 90E         ?C#0   P                Standard format?
1082 18274 16          GOYES  STDF30           No.  Fix, Sci or Eng.
1083 18276 95C         ?A#0   M                Number=0?
1084 18279 A0          GOYES  FNDS05           No.
1085 1827B 3103        LCASC  \0\              Yes.  Display as "0".
1086 1827F 6482        GOTO   STKCHR
1087 18283 05   FNDS05 SETDEC
1088 18285 3221        LCHEX  012
           0
1089 1828A D5          B=C    A
1090 1828C 22          P=     2                Find LSD.  Start at bottom.
1091 1828E 0C   FNDS10 P=P+1
1092 18290 CD          B=B-1  A
```

```
1093 18292 908          ?A=0   P          Is this digit zero?
1094 18295 9F           GOYES  FNDS10     Yes, look at next one.
1095 18297 0C           P=P+1              No.
1096 18299 80FF         CPEX   15
1097 1829D 04           SETHEX
1098 1829F BCE          C=-C-1 S          C[S]=B[X]=# sig digits - 1.
1099 182A2 05           SETDEC
1100 182A4 20           P=     0
1101 182A6 B38          B=B-A  X          #Sig dig - exponent
1102 182A9 581          GONC   STDF10     B[X]=dspdgt (if not carry)
1103 182AC 3200         LCHEX  500        Why did we carry?
          5
1104 182B1 9B6          ?A>C   X          Exponent negative?
1105 182B4 42           GOYES  NEGEXP     Yes.  Process it.
1106 182B6 3211         LCHEX  011        No.
          0
1107 182BB 9B6          ?A>C   X          Exponent > 11?
1108 182BE 42           GOYES  SCINj       Yes, sci notation
1109 182C0 D1           B=0    A          No, must be FIX 0
1110 182C2 2F    STDF10 P=     15         B[X] = dspdgt
1111 182C4 0C    STDF20 P=P+1             Loop to convert to HEX
1112 182C6 A3D          B=B-1  X
1113 182C9 5AF          GONC   STDF20
1114 182CC 80F1         CPEX   1
1115 182D0 20           P=     0
1116 182D2 309          LCHEX  9          Dope up FIX display
1117 182D5 441   STDF30 GOC    FXSCEN     B.E.T.
1118 182D8 3221  NEGEXP LCHEX  012
          0
1119 182DD 9BD          ?B<=C  X          FIX >12?
1120 182E0 2E           GOYES  STDF10     No.  Display FIX.
1121 182E2 812   SCINj  CSLC              Display SCI.  Q to C[0].
1122 182E5 F2           CSL    A
1123 182E7 302          LCHEX  2          Display SCI Q.  Fall through.
```

```
1124                    STITLE FXSCEN - Output in FIX, SCI or ENG
1125                ****************************************************************
1126                ****************************************************************
1127                **
1128                ** Name:    FXSCEN  -  Output number in fix/sci/eng fmt
1129                **
1130                ** Category:   LOCAL
1131                **
1132                ** Purpose:
1133                **      Part of CLCSTR used to output number in FIX, SCI or ENG
1134                **      format.
1135                **
1136                ** Entry:
1137                **      Number to be displayed in A.
1138                **      Mantissa sign has already been written to string.
1139                **      R1[A] points to beginning of string in stack (used to
1140                **        make string header later).
1141                **      D[A]=(AVMEMS).
1142                **      D1 points to current top of stack.
1143                **      C[B] = display flags:
1144                **        Bits 0-1: 1=fix, 2=sci, 3=eng.
1145                **        Bit 3: 1 if FIX display called from std fmt code
1146                **               (Causes 0 before dp to be suppressed if
1147                **                exponent negative and dp to be suppressed
1148                **                if no fractional part.  This bit should only
1149                **                be set only if number is displayed in FIX
1150                **                format.)
1151                **        Bits 4-7: Display setting (0-12).
1152                **      P=0.
1153                **
1154                ** Exit:
1155                **      Carry set.
1156                **      Mantissa, exponent sign and exponent written to string
1157                **        on stack.
1158                **      D1 points to current top-of-stack.
1159                **      R1[A] unchanged.
1160                **      P=0.
1161                **
1162                ** Calls:      STKCHR.
1163                **
1164                ** Uses.......
1165                **              A,B,C,D[15-13],D1,P
1166                **
1167                ** Stk lvls:   1
1168                **
1169                ** Detail:
1170                **      Converts number to "display format", after which
1171                **      TOASCI outputs characters to stack.  Display format
1172                **      looks as follows:
1173                **        Mantissa is in B:
1174                **          B[14] through first "F" are digits.
1175                **          B[0] is DP position...there are B[0]+1 digits
1176                **            before the DP.  B[0]=F means DP is not written
1177                **            or (during TOASCI looping) has already been
1178                **            written.
```

```
1179                **        Exponent is in A:
1180                **          A[S] is sign: 0 for "+", 1 for "-".
1181                **          A[14] through first "F" are digits.
1182                **          A=FFFFFFFFFFFFFFF means no exponent.
1183                **
1184                ** History:
1185                **
1186                **    Date     Programmer            Modification
1187                **  --------   ----------   ------------------------------
1188                **  10/21/82   NM           Added documentation
1189                **
1190                ********************************************************************
1191                ********************************************************************
1192                *-
1193                *- Note that above code falls through to here
1194                *-
1195 182EA 04    FXSCEN SETHEX
1196 182EC BF3         DSL    W
1197 182EF BF3         DSL    W
1198 182F2 BF3         DSL    W
1199 182F5 AE7         D=C    B              Hold DSPFLG byte
1200 182F8 21          P=     1
1201 182FA 30C         LC(1)  12             Maximum display setting.
1202 182FD 98F         ?D<=C  P              Setting legal?
1203 18300 50          GOYES  FXSC10         Yes.
1204 18302 A87         D=C    P              No.  Use 12.
1205 18305 AF1   FXSC10 B=0    W
1206 18308 ADC         ABEX   M              Split exponent and mantissa
1207 1830B AA6         C=A    XS
1208 1830E AC0         A=0    S              To hold exponent sign
1209 18311 05          SETDEC
1210 18313 A26         C=C+C  XS             Exponent negative?
1211 18316 550         GONC   DSPM20         No
1212 18319 B44         A=A+1  S              Mark exponent sign negative
1213 1831C AEB   DSPM20 C=D    B
1214 1831F 0B          CSTEX                 Dspflg to ST
1215 18321 871         ?ST=1  1              Sci/Eng?
1216 18324 60          GOYES  SCI            Yes
1217 18326 61C0        GOTO   FIX            No.  Fix.
1218 1832A 0B    SCI   CSTEX                 Restore original ST
1219 1832C 04    SCI03 SETHEX
1220 1832E 21          P=     1
1221 18330 30D         LCHEX  D
1222 18333 B0B         C=C-D  P              Lopping position
1223 18336 80D1        P=C    1
1224 1833A 05          SETDEC
1225 1833C AF2         C=0    W
1226 1833F A99         C=B    WP
1227 18342 A71         B=C+B  W              Do rounding
1228 18345 949         ?B=0   S              Rounding overflow?
1229 18348 E0          GOYES  SCI05          No
1230 1834A BF5         BSR    W              Yes.  sr mantissa.
1231 1834D B34         A=A+1  X              Increment exponent
1232 18350 550         GONC   SCI05          Exponent now positive?
1233 18353 AC0         A=0    S              Yes, change sign
```

```
1234 18356 A91   SCI05  B=0    WP          Lop
1235 18359 04           SETHEX
1236 1835B A1D          B=B-1  WP          Trailing f's
1237 1835E 20           P=     0
1238 18360 A81          B=0    P           Zero for DP position
1239 18363 05           SETDEC
1240 18365 948          ?A=0   S           Exponent positive?
1241 18368 F0           GOYES  SCI10       Yes
1242 1836A BB8          A=-A   X           Negate exponent
1243 1836D 3299         LCHEX  499
             4
1244 18372 932          ?A=C   X           Exponent=-499?
1245 18375 A5           GOYES  SCI20       Yes.  Display SCI (not ENG)
1246 18377 AEB   SCI10  C=D    B
1247 1837A 0B           CSTEX
1248 1837C 870          ?ST=1  0           Sci or Eng?
1249 1837F 20           GOYES  SCI15       Carry set if Eng
1250 18381 0B    SCI15  CSTEX
1251 18383 5B4          GONC   SCI20
1252 18386 3200         LCHEX  300
             3
1253 1838B AB7          D=C    X
1254 1838E D6           C=A    A           Prepare for mod loop
1255 18390 B3B   ENG10  C=C-D  X           Loop to compute exp mod 3
1256 18393 5CF          GONC   ENG10
1257 18396 A3B          C=C+D  X
1258 18399 BB7          DSR    X           Shift divisor
1259 1839C 93F          ?D#0   X           Done?
1260 1839F 1F           GOYES  ENG10       No
1261 183A1 93A          ?C=0   X           Exponent divisible by 3?
1262 183A4 B2           GOYES  SCI20       Yes
1263 183A6 A85          B=C    P           No.  new dp position.
1264 183A9 EA           A=A-C  A           New exponent
1265 183AB 948          ?A=0   S           Exponent positive?
1266 183AE A0           GOYES  ENG20       Yes
1267 183B0 303          LCHEX  3
1268 183B3 CA           A=A+C  A           New exponent
1269 183B5 B0D          B=C-B  P           New dp position
1270 183B8 04    ENG20  SETHEX
1271 183BA 30D          LCHEX  D
1272 183BD B09          C=C-B  P
1273 183C0 80D0         P=C    0           Point at digit just below dp
1274 183C4 0C    ENG30  P=P+1
1275 183C6 B05          B=B+1  P           Need zeroes to reach dp?
1276 183C9 4AF          GOC    ENG30       Yes
1277 183CC A0D          B=B-1  P           Done inserting zeroes
1278 183CF 04    SCI20  SETHEX
1279 183D1 E4           A=A+1  A           Hex increment of exp
1280 183D3 2E           P=     14
1281 183D5 B90   SCI30  ASL    WP          Loop to align exp w/sign
1282 183D8 908          ?A=0   P           Aligned?
1283 183DB AF           GOYES  SCI30       No
1284 183DD A1C          A=A-1  WP          Yes.  restore exp & put F's
1285 183E0 6F90         GOTO   TOASCI
1286 183E4 674F  FSCI   GOTO   SCI03       .
```

```
1287 183E8 840   FIX    ST=0   0          Clear so doesn't appear eng
1288 183EB 0B           CSTEX
1289 183ED AE7          D=C    B
1290 183F0 80D1         P=C    1
1291 183F4 D6           C=A    A          Exponent
1292 183F6 E6    FIX10  C=C+1  A          Add dec(dspsetting)+1 to exp
1293 183F8 0D           P=P-1
1294 183FA 5BF          GONC   FIX10
1295 183FD A26          C=C+C  XS         Result negative?
1296 18400 43E          GOC    FSCI       Yes.  display in sci.
1297 18403 0D    FIX20  P=P-1              Loop to find rounding digit
1298 18405 891          ?P=    1          Overflow display?
1299 18408 63           GOYES  FSCI?      Yes.  May need SCI display.
1300 1840A A3E          C=C-1  X          No
1301 1840D 55F          GONC   FIX20
1302 18410 AF2          C=0    W
1303 18413 959          ?B=0   M          Mantissa=0?
1304 18416 93           GOYES  FIX35      Yes.  Don't round.
1305 18418 A99          C=B    WP
1306 1841B A79          C=C+B  W
1307 1841E A92          C=0    WP         Perform rounding
1308 18421 97A          ?C=0   W          Number rounds to zero?
1309 18424 0C           GOYES  FSCI       Yes, display in SCI.
1310 18426 94A          ?C=0   S          Rounding overflow?
1311 18429 62           GOYES  FIX35      No
1312 1842B BF6          CSR    W          Yes.  sr mantissa.
1313 1842E B34          A=A+1  X          And increment exponent.
1314 18431 550          GONC   FIX27      Exponent went positive?
1315 18434 AC0          A=0    S          Yes.
1316 18437 0D    FIX27  P=P-1              Move lopping pointer
1317 18439 881          ?P#    1          Too many digits now?
1318 1843C 31           GOYES  FIX35      No.
1319 1843E 20    FSCI?  P=     0          Can't display full FIX.
1320 18440 3211         LCHEX  011        Perhaps we can ignore trlg 0's.
          0
1321 18445 22           P=     2          For lopping
1322 18447 9B6          ?A>C   X          Exponent>11?
1323 1844A A9           GOYES  FSCI       Yes.  MUST display SCI.
1324 1844C AF9          C=B    W          No.  Will knock off trlng 0's.
1325 1844F AF5   FIX35  B=C    W          Copy new mantissa
1326 18452 948          ?A=0   S          Exponent negative?
1327 18455 D0           GOYES  FIX55      No.
1328 18457 BF5   FIX40  BSR    W          Loop to shift in leading 0's
1329 1845A 0D           P=P-1              And move lopping pointer.
1330 1845C B34          A=A+1  X          Increment exponent.
1331 1845F 57F          GONC   FIX40      Go if need more zeros.
1332 18462 04    FIX55  SETHEX
1333 18464 A1D          B=B-1  WP         Trailing F's
1334 18467 20           P=     0
1335 18469 A88          B=A    P          Lonib of exponent
1336 1846C BE4          ASR    B          Hinib of exponent
1337 1846F 908          ?A=0   P          Hinib=0?
1338 18472 80           GOYES  FIX60      Yes
1339 18474 30A          LCHEX  A          No.  =1
1340 18477 A01          B=B+C  P          DP position.
```

```
1341 1847A AF0   FIX60  A=0    W
1342 1847D A7C          A=A-1  W              F's for exponent
1343 18480 20    TOASCI P=     0              Encode in ascii
1344 18482 A84          A=B    P              Dp pointer to a
1345 18485 A07          D=D+D  P              Set carry if STD-fixed fmt
1346 18488 BF7          DSR    W
1347 1848B BF7          DSR    W
1348 1848E BF7          DSR    W              Restore avmems
1349 18491 BF1          BSL    W              Position first digit for output
1350 18494 5F1          GONC   TOAS10         Go if not STD-fixed
1351 18497 949          ?B=0   S              Leading zero?
1352 1849A D2           GOYES  TOAS17         Yes.  DP is next.
1353 1849C 30E          LCHEX  E
1354 1849F B02          C=C-A  P
1355 184A2 80D0         P=C    0              Point at digit past DP.
1356 184A6 30F          LCHEX  F
1357 184A9 905          ?B#C   P              Blank?
1358 184AC 60           GOYES  TOAS05         No.
1359 184AE D0           A=0    A
1360 184B0 CC           A=A-1  A              Suppress DP.
1361 184B2 20    TOAS05 P=     0
1362 184B4 AC9   TOAS10 C=B    S              Get current digit
1363 184B7 B45          B=B+1  S              Done with mantissa?
1364 184BA 4F1          GOC    TOAS20         Yes
1365 184BD 303          LCHEX  3              No.
1366 184C0 812          CSLC                  Create ascii digit
1367 184C3 7D30  TOAS15 GOSUB  STKCHR         Output digit
1368 184C7 BF1   TOAS17 BSL    W              Shift in next digit
1369 184CA A0C          A=A-1  P              At dp?
1370 184CD 56E          GONC   TOAS10         No
1371 184D0 31E2         LCASC  \.\            Yes
1372 184D4 BF5          BSR    W              So won't sl extra digit
1373 184D7 4BE          GOC    TOAS15         Bet
1374 184DA B44   TOAS20 A=A+1  S              Any exponent?
1375 184DD 400          RTNC                  No
1376 184E0 3154         LCASC  \E\            Yes
1377 184E4 7C10         GOSUB  STKCHR         Write "E"
1378 184E8 A4C          A=A-1  S
1379 184EB 948          ?A=0   S              Exponent sign positive?
1380 184EE A0           GOYES  TOAS30         Yes
1381 184F0 31D2         LCASC  \-\
1382 184F4 7C00         GOSUB  STKCHR         No, write out sign
1383 184F8 AF8   TOAS30 B=A    W              Put exponent where mantissa was
1384 184FB AF0          A=0    W
1385 184FE A7C          A=A-1  W              F's for exponent
1386 18501 45C          GOC    TOAS17         Write out exponent.  B.E.T.
1387            *-
1388            *-
```

```
1389                    STITLE STKCHR - Add character to stack
1390                ****************************************************************
1391                ****************************************************************
1392                **
1393                ** Name:(S) STKCHR  -  Add a Character to a Stack Item
1394                ** Name:    STKCH+  -  Add a Character to a Stack Item
1395                **
1396                ** Category:   MTHUTL
1397                **
1398                ** Purpose:
1399                **      Decrements stack pointer, checking av mem to be sure
1400                **      enough room exists.  Character C(B) is then written
1401                **      to memory.  STKCH+ is same except doesn't move stack
1402                **      pointer first.
1403                **
1404                ** Entry:
1405                **      C(B)=Character to be appended to stack
1406                **      D(A)=(AVMEMS)
1407                **      D1 points to stack
1408                **
1409                ** Exit:
1410                **      Exits to MFERR with eMEM error if not enough room
1411                **      D1 points to new stack character
1412                **      Carry clear.
1413                **
1414                ** Calls:      None
1415                **
1416                ** Uses.......
1417                **  Inclusive: D1
1418                **
1419                ** Stk lvls:   0
1420                **
1421                ** History:
1422                **
1423                **    Date     Programmer             Modification
1424                **  --------   ----------   ----------------------------------
1425                **  07/20/82   B.S.         Updated documentation
1426                **
1427                ****************************************************************
1428                ****************************************************************
1429 18504 1C1  =STKCHR D1=D1- 2
1430 18507 137  =STKCH+ CD1EX
1431 1850A 8B3          ?C<D   A
1432 1850D 71           GOYES  STKC10
1433 1850F 137          CD1EX
1434 18512 14D          DAT1=C B
1435 18515 03           RTNCC
1436                *-
1437                *-
1438                *   C(A) is added to A(A).  This is then subtracted from
1439                *     (AVMEME) and if this is past (AVMEMS) then jumps to
1440                *     MEMERR else returns
1441 18517 CA    CHKMEM A=A+C  A
1442 18519 7031         GOSUB  GETAVM         D1=AVE,D=AVMS
1443 1851D E2           C=C-A  A
```

```
1444 1851F 8BF          ?C>=D  A
1445 18522 00           RTNYES
1446 18524 6B5A  STKC10 GOTO   MEMERj
1447            *-
1448            *-
1449 18528 8C00 =EXPEXj GOLONG =expexc
           00
1450 1852E 8C00 =popmth GOLONG =POPMTH
           00
1451
```

```
1452                    STITLE CKINFO - Set up device info
1453            ****************************************************************
1454            ****************************************************************
1455            **
1456            ** Name:(S) CKINFO  -  Check Handler Information
1457            ** Name:(S) CKINF-  -  Specify DISP Stmt & Set Handler Info
1458            **
1459            ** Category:   EXCUTL
1460            **
1461            ** Purpose:
1462            **      Guarantees that info in STMTR0,STMTR1 is correct for
1463            **      the statement that is being executed.
1464            **
1465            ** Entry:
1466            **      P=0,HEXMODE
1467            **
1468            ** Exit:
1469            **      P=0,Carry clear
1470            **
1471            ** Calls:      POLL
1472            **
1473            ** Uses.......
1474            **  Exclusive: A,  C
1475            **  Inclusive: A,B,C,D,FUNCD0,FUNCD1,FUNCR0,FUNCR1,STMTR0
1476            **
1477            ** Stk lvls:   <4
1478            **
1479            ** NOTE:
1480            **      Function RAM is NOT preserved through CKINFO!!!
1481            **
1482            **      If MLFFLG is not clear, MLFFLG,STMTR0 and STMTR1 are
1483            **      updated
1484            **
1485            ** Detail:
1486            **      RAM utilization:
1487            **      ----------------
1488            **            Pos/Width  EOLLEN (Number of nibs)
1489            **                   |      |
1490            **        Handler    |      | Char#1
1491            **           |       |      |  |
1492            **  Statement|       |      |  |  Char#2
1493            **       Type|       |      |  |  |
1494            **         | |       |      |  |  |  Char#3
1495            **  MLFFLG | |       |      |  |  |  |
1496            **       | | |       |      |  |  |  |  Reserved for polled
1497            **       v v v       v      v  v  v  v  v             handlers
1498            **      ---------------------------------------------------
1499            **      |1|1|  5  |  5  |1|2 |2 |2 |        14         |
1500            **      ---------------------------------------------------
1501            **         ^                        ^
1502            **         |                        |
1503            **       STMTR0                   STMTR1
1504            **
1505            **      If MLFFLG is clear then routine returns quickly
1506            **      otherwise a handler address and other information
```

```
1507            **      is set up to transfer information to the device
1508            **      which is appropriate for the statement.  The states
1509            **      are coded as follows:
1510            **
1511            **      MLFFLG          0 --> Information okay
1512            **                      F --> Information not reliable
1513            **
1514            **      Statement type  0 --> DISP
1515            **                      1 --> PRINT
1516            **                      2-F --> POLL for setup
1517            **                                2 --> OUTPUT
1518            **                                3 --> PLOT
1519            **                                4-F --> Reserved
1520            **
1521            ** History:
1522            **
1523            **    Date     Programmer              Modification
1524            **  --------   ----------   ----------------------------------
1525            **  11/09/82   N.Z.         Updated documentation
1526            **
1527            ********************************************************************
1528            ********************************************************************
1529            ********************************************************************
1530            ********************************************************************
1531            **
1532            ** Name:(S) pPRTIS  -  PRINTER IS handler poll
1533            **
1534            ** Category:   POLL
1535            **
1536            ** Type:       POLL
1537            **
1538            ** Purpose:
1539            **      Set up for the PRINT statement and return the address
1540            **      of a handler for the print items.
1541            **
1542            ** Should poll be "Handled" (return with XM=0)?:
1543            **      YES
1544            **
1545            ** Meaning of "Handling" Poll (what does code do if handled?):
1546            **      A handler for the PRINT statement has been provided and
1547            **      its address returned.
1548            **
1549            ** Entry conditions for handler (registers, ST, RAM, etc.):
1550            **      Carry clear
1551            **      B[A] = Poll number.
1552            **      HEX mode.
1553            **      P=0.
1554            **
1555            ** Normal exit conditions from handler if handled (ST, RAM,
1556            ** registers, etc.):
1557            **      Carry clear
1558            **      A(A) is the address of the PRINT handler
1559            **      HEX mode.
1560            **      XM=0.
1561            **
```

```
1562            ** Normal exit conditions from handler if not handled (ST, RAM,
1563            ** registers, etc.):
1564            **      Carry clear (POLL only).
1565            **      HEX mode.
1566            **      XM=1.
1567            **
1568            ** Error exit conditions from handler (POLL only):
1569            **      Not applicable
1570            **
1571            ** Available subroutine levels:
1572            **      4
1573            **
1574            ** NOTE:
1575            **      This poll is issued in the CKINFO routine which is in
1576            **      the process of setting up statement scratch to handle
1577            **      a PRINT/PLIST statements output.
1578            **
1579            ** What registers/RAM may be used if handled?:
1580            **      Must not alter D1 or any status bits or any R registers
1581            **      Function scratch is available
1582            **
1583            ** What registers/RAM may be used if not handled?:
1584            **      A-D, D0, D1, P
1585            **
1586            ** What registers/RAM may be used if error exit (POLL only)?:
1587            **      Not applicable
1588            **
1589            ** Special memory/pointer considerations (are pointers funny?):
1590            **      Normal memory configuration
1591            **
1592            ** Envisioned application(s):
1593            **      Extend PRINT/PLIST commands to handle unknown
1594            **      destination devices (specifically HPIL devices)
1595            **
1596            ** History:
1597            **
1598            **    Date     Programmer            Modification
1599            ** --------   ----------   ---------------------------------
1600            **  11/09/82   N.Z.         Added documentation
1601            **  10/17/83   B.S.         Updated documentation
1602            **
1603            ****************************************************************
1604            ****************************************************************
1605            ****************************************************************
1606            ****************************************************************
1607            **
1608            ** Name:(S) pPRTCL  -  PRINT class statement handler poll
1609            **
1610            ** Category:   POLL
1611            **
1612            ** Type:       POLL
1613            **
1614            ** Purpose:
1615            **      Set up a handler for a statement type not recognized
1616            **      by the mainframe.
```

```
1617            **
1618            ** Should poll be "Handled" (return with XM=0)?:
1619            **      Yes
1620            **
1621            ** Meaning of "Handling" Poll (what does code do if handled?):
1622            **      The statement type has been recognized and statement
1623            **      scratch has been set up in accordance with CKINFO
1624            **      specifications for the specified type of statement.
1625            **
1626            ** Entry conditions for handler (registers, ST, RAM, etc.):
1627            **      Carry clear
1628            **      B[A] = Poll number.
1629            **      HEX mode.
1630            **      First nib of STMTR0 is statement type
1631            **      P=0.
1632            **
1633            ** Normal exit conditions from handler if handled (ST, RAM,
1634            ** registers, etc.):
1635            **      Carry clear
1636            **      HEX mode.
1637            **      XM=0.
1638            **      STMTR0,STMTR1 set according to CKINFO specifications
1639            **
1640            ** Normal exit conditions from handler if not handled (ST, RAM,
1641            ** registers, etc.):
1642            **      Carry clear
1643            **      HEX mode.
1644            **      XM=1.
1645            **
1646            ** Error exit conditions from handler (POLL only):
1647            **      Not applicable
1648            **
1649            ** Available subroutine levels:
1650            **      4
1651            **
1652            ** NOTE:
1653            **      Function scratch is available
1654            **      SCRTCH, SNAPBF, TRFMBF, LDCSPC
1655            **
1656            ** What registers/RAM may be used if handled?:
1657            **      Statement scratch should be set by poll handler
1658            **      A-D, D0, P
1659            **
1660            ** What registers/RAM may be used if not handled?:
1661            **      A-D, D0, D1, P
1662            **
1663            ** What registers/RAM may be used if error exit?:
1664            **      Not applicable
1665            **
1666            ** Special memory/pointer considerations (are pointers funny?):
1667            **      No special conciderations
1668            **
1669            ** Envisioned application(s):
1670            **      Allows adding new keywords in the same class as DISP
1671            **      and PRINT.
```

```
1672                **
1673                ** History:
1674                **
1675                **    Date     Programmer          Modification
1676                **  --------   ----------   ----------------------------------
1677                **  11/09/82   N.Z.         Added documentation
1678                **  10/18/83   B.S.         Updated documentation
1679                **
1680                ********************************************************************
1681                ********************************************************************
1682 18534 31F0   =CKINF- LC(2)  (DISPt)*16+#F
1683 18538 1F00    CKINF+ D1=(5) =MLFFLG
           000
1684 1853F 14D           DAT1=C  B
1685 18542 136     =CKINFO CDOEX
1686 18545 06            RSTK=C                  Save D0 on RSTK
1687 18547 1B00          D0=(5) =MLFFLG
           000
1688 1854E 14E           C=DAT0 B                Read MultiLine Function FLaG
1689 18551 90A           ?C=0   P                Is everything okay?
1690 18554 63            GOYES  CKIN10           Yes, then return
1691 18556 BB6           CSR    X                Get statement type
1692 18559 1542          DAT0=C XS               Clear MultiLine Function Flag
1693 1855D A0E           C=C-1  P                DISP statement?
1694 18560 503           GONC   CKIN20           No, look for other stmt types
1695 18563 39      CKIN03 NIBHEX 39              LC(10)
1696 18565 0000          CON(5) =DPOS              DPOS/WIDTH pointer
           0
1697 1856A 4D0A          NIBHEX 4D0A0              EOL string = CR/LF (length=2)
           0
1698 1856F AFA           A=C    W                Move to A
1699 18572 343A          LC(5)  =DPART2
           E71
1700 18579 1B00    CKIN05 D0=(5) (=STMTR0)+6     Point to POS/WIDTH pointer
           000
1701 18580 158B          DAT0=A 12               Write POS/WIDTH address and
1702                                               EOL string
1703 18584 184           D0=D0- 5                Move pointer back to
1704                                               Format/Handler address
1705 18587 144           DAT0=C A
1706 1858A 07      CKIN10 C=RSTK                 Pop off original D0
1707 1858C 134           D0=C                    Restore D0
1708 1858F 03            RTNCC                   Return with carry clear
1709                *-
1710                *-
1711 18591 A0E     CKIN20 C=C-1  P               PRINT statement?
1712 18594 513           GONC   CKIN30           No, then continue search
1713 18597 8E00          GOSUBL =POLL            Poll all ROMS for PRINT handler
           00
1714 1859D F0            CON(2) =pPRTIS          Process number for POLL
1715 1859F 831           ?XM=0                   Was poll handled?
1716 185A2 B0            GOYES  CKIN25           Yes, then okay
1717 185A4 343A          LC(5)  =DPART2          No, then use DPART2
           E71
1718 185AB DA            A=C    A
```

```
1719                *Handler now in A(A)
1720 185AD 1B00    CKIN25 DO=(5) (=EOLLEN)-5    Position to read into A[15-5]
           000
1721 185B4 15EB           C=DATO 12
1722 185B8 3400           LC(5)  =PPOS          Print position/width address
           000
1723 185BF AFE            ACEX   W
1724 185C2 66BF           GOTO   CKIN05         Set info into memory
1725                *-
1726 185C6         CKIN30
1727                * At this point all known statement types have been handled.
1728                * We now do a poll to see if any other type of statement of
1729                * printer statement class called this routine and would like a
1730                * chance to set up a handler.
1731 185C6 8E00           GOSUBL =POLL
           00
1732 185CC E0             CON(2) =pPRTCL        Print Class statement poll
1733 185CE 831            ?XM=0
1734 185D1 9B             GOYES  CKIN10         Return from CKINFO...handled
1735                * Not handled...default to DISP statement
1736 185D3 5F8            GONC   CKIN03         B.E.T.
1737                *-
1738                *-
```

```
1739                    STITLE DSP$00 - DISP$ function
1740                ******************************************************************
1741                ******************************************************************
1742                **
1743                ** Name:(S) DSP$00  -  Create String of Readable Characters
1744                **
1745                ** Category:   DSPUTL
1746                **
1747                ** Purpose:
1748                **      Adds a string to stack containing all readable chars
1749                **      in display buffer.
1750                **
1751                ** Entry:
1752                **      P      = 0
1753                **      D1 points to top of stack
1754                **      S0 set implies append CR and RTN when done.
1755                **      S0 clear implies no CR on end and jump to EXPR when
1756                **          done.
1757                **
1758                ** Exit:
1759                **      P      = 0
1760                **      D1 points to new string on top of stack
1761                **      If Return(S0) set then CR will have been appended
1762                **      Exits to EXPR if S0 clear.
1763                **
1764                ** Calls:      STKCHR,ADHEAD
1765                **
1766                ** Uses.......
1767                **  Exclusive: R1,D1,A(A),B(W),C(14-0),D(A)
1768                **  Inclusive: R1,D1,A(A),B(W),C(W),D(A)
1769                **
1770                ** Stk lvls:   3
1771                **
1772                ** Detail:
1773                **      Examines display buffer and copys all "unprotected"
1774                **      characters into a string on the math stack.  If S0
1775                **      is set then a CR is appended following the last char
1776                **      in the string.  A standard string header is attached
1777                **      with D1 pointing to it.  If S0 is clear then the
1778                **      routine will jump to EXPR to continue expression
1779                **      execute instead of returning.
1780                **
1781                ** History:
1782                **
1783                **    Date     Programmer             Modification
1784                **  --------   ----------   ----------------------------------
1785                **  07/20/82   B.S.         Updated documentation
1786                **
1787                ******************************************************************
1788                ******************************************************************
1789 185D6 00            NIBHEX 00
1790 185D8 840  =DSP$    ST=0   Return          Don't return when done
1791 185DB 136  =DSP$00  CDOEX
1792 185DE 06            RSTK=C                 Save D0
1793 185E0 133           AD1EX
```

```
1794 185E3 131          D1=A                  Copy stack pointer to C
1795 185E6 76A0         GOSUB  d=avs+         Save stack ptr in R1
1796                                          D(A)=(AVMEMS)
1797 185EA 1B00         DO=(5) (=DSPMSK)+12   Point to dsply mask (1st 48 bits)
           000
1798 185F1 1565         C=DATO M              Read in 48 bits
1799 185F5 1A00         DO=(4) =DSPBFS        Start of display buffer
           00
1800 185FB 7A20         GOSUB  DSP$30         Process first 48 characters
1801 185FF 461          GOC    DSP$10         If null found then finish up
1802 18602 1A00         DO=(4) =DSPMSK        Point to dsply mask(2nd 48 bits)
           00
1803 18608 1565         C=DATO M              Read in 48 bits
1804 1860C 1A00         DO=(4) (=DSPBFS)+2*48 Point to second half of display
           00
1805 18612 7310         GOSUB  DSP$30         Process second 48 characters
1806 18616 860   DSP$10 ?ST=0  Return
1807 18619 A0           GOYES  DSP$20
1808 1861B 31D0         LCHEX  0D             Carriage return
1809 1861F 71EE         GOSUB  STKCHR         Append CR
1810 18623 07    DSP$20 C=RSTK
1811 18625 6360         GOTO   BF2S30         Append string header
1812             *-
1813             *-
1814 18629 31F2  DSP$30 LC(2)  47
1815 1862D AF5          B=C    W              Copy bitmask to B(M)
1816                                          Count to B(B)
1817 18630 14E   DSP$40 C=DATO B              Read a byte
1818 18633 161          DO=DO+ 2              Skip past that byte
1819 18636 96A          ?C=0   B              Is it a null byte?
1820 18639 00           RTNYES                Yes, then return immediately
1821 1863B A55          B=B+B  M              Check for readable/unreadable
1822 1863E 460          GOC    DSP$50         Skip output if unreadable
1823 18641 7FBE         GOSUB  STKCHR         Stack character
1824 18645 A6D   DSP$50 B=B-1  B              Decrement count
1825 18648 57E          GONC   DSP$40         Loop back until done
1826 1864B 03           RTNCC                 Clear carry indicates
1827                                            normal termination
1828             *-
1829             *-
```

```
1830                    STITLE GETAVM - Get (AVMEME),(AVMEavail mem ptrs
1831            ****************************************************************
1832            ****************************************************************
1833            **
1834            ** Name:(S) GETAVM  -  Get Available memory limits
1835            **
1836            ** Category:   PTRUTL
1837            **
1838            ** Purpose:
1839            **      Reads (AVMEME) into C & D1 and (AVMEMS) into D(A)
1840            **
1841            ** Entry:
1842            **
1843            ** Exit:
1844            **      D(A)    =  (AVMEMS)
1845            **      C(A),D1 =  (AVMEME)
1846            **
1847            ** Calls:      D=AVMS
1848            **
1849            ** Uses.......
1850            **  Inclusive: C(A),D(A),D1
1851            **
1852            ** Stk lvls:   1
1853            **
1854            ** History:
1855            **
1856            **    Date     Programmer              Modification
1857            **  --------   ----------   ----------------------------------
1858            **  10/18/83   B.S.         Updated documentation
1859            **
1860            ****************************************************************
1861            ****************************************************************
1862            ****************************************************************
1863            ****************************************************************
1864            **
1865            ** Name:(S) D1=AVE  -  Set D1 to (AVMEME)
1866            **
1867            ** Category:   PTRUTL
1868            **
1869            ** Purpose:
1870            **      Reads (AVMEME) into D1 (and C(A))
1871            **
1872            ** Entry:
1873            **
1874            ** Exit:
1875            **      D1,C(A) = (AVMEME)
1876            **
1877            ** Calls:      None
1878            **
1879            ** Uses.......
1880            **  Inclusive: C(A),D1
1881            **
1882            ** Stk lvls:   0
1883            **
1884            ** History:
```

```
1885                **
1886                **    Date     Programmer              Modification
1887                **  --------   ----------   ----------------------------------
1888                **  10/18/83   B.S.         Added Documentation
1889                **
1890                ****************************************************************
1891                ****************************************************************
1892 1864D 7240   =GETAVM GOSUB  d=avms
1893 18651 1F00   =D1=AVE D1=(5) =AVMEME
           000
1894 18658 147     D1=@D1 C=DAT1 A
1895 1865B 135            D1=C
1896 1865E 03             RTNCC
1897                *-
1898                *-
```

```
1899                    STITLE BF2STK - Buffer to stack
1900                ****************************************************************
1901                ****************************************************************
1902                **
1903                ** Name:(S) BF2STK  -  Buffer To Stack
1904                ** Name:    BF2ST+  -  Buffer To Stack
1905                **
1906                ** Category:   MTHSTK
1907                **
1908                ** Purpose:
1909                **      Pushes a string buffer onto math stack
1910                **
1911                ** Entry:
1912                **      P      = 0
1913                **      S0 = 0 ---> GOTO EXPR when done (don't return)
1914                **      S0 = 1 ---> Return when done
1915                **      BF2ST+ pre-clears S0 causing a GOTO EXPR when done
1916                **      D1 points to stack
1917                **      D0 should be PC if S0 clear for proper function rtn
1918                **      C(A) should point to buffer which is a string of
1919                **      bytes terminated by a FF byte.
1920                **
1921                ** Exit:
1922                **      P      = 0
1923                **      D1 reflects new stack pointer
1924                **      D0 unchanged
1925                **
1926                ** Calls:      STKCHR,ADHEAD,D=AVMS
1927                **
1928                ** Uses.......
1929                **  Inclusive: A(A),B(A),C(A),D(A),R1,D0,D1
1930                **
1931                ** Stk lvls:   1
1932                **
1933                ** Detail:
1934                **      Buffer is terminated by an FF byte.
1935                **      Pushes a buffer onto stack a character at a time
1936                **      and jumps to MEMERR if memory overflows.  The result
1937                **      is a string item on stack with proper header set up.
1938                **      If S0 is clear the routine assumes that a function
1939                **      is ending returns directly to EXPR to continue
1940                **      expression evaluation.
1941                **
1942                ** History:
1943                **
1944                **    Date     Programmer             Modification
1945                **  --------   ----------   ----------------------------------
1946                **  10/19/82   B.S.         Updated documentation
1947                **
1948                ****************************************************************
1949                ****************************************************************
1950 18660 840  =BF2ST+ ST=0   Return         Don't return (goto EXPR)
1951 18663 136  =BF2STK CD0EX
1952 18666 D5           B=C    A              Save D0
1953 18668 133          AD1EX
```

```
1954 1866B 7120          GOSUB  d=avs+          Save stack start
1955                                            D(A)=(AVMEMS) for STKCHR
1956 1866F 131           D1=A                   Restore stack pointer
1957 18672 14A   BF2S10  A=DAT0 B               Read character
1958 18675 D6            C=A    A               Make a copy
1959 18677 B64           A=A+1  B               Is it an FF byte?
1960 1867A 4C0           GOC    BF2S20          Yes, then done
1961 1867D 738E          GOSUB  STKCHR          No, put char on stack
1962 18681 161           D0=D0+ 2               Move to next character
1963 18684 5DE           GONC   BF2S10          (B.E.T.)
1964 18687 D9    BF2S20  C=B    A
1965 18689 134   BF2S30  D0=C                   Restore D0
1966 1868C 6A2B  adhead  GOTO   ADHEAD          Put string header on stack
1967                                              and finish up
1968             *-
1969             *-
1970 18690 101    d=avs+ R1=A
1971 18693 8C00   d=avms GOLONG =D=AVMS
           00
```

```
1972                    $TITLE INPUT - Statement execution
1973            ********************************************************************
1974            ********************************************************************
1975            **
1976            ** Name:    INPUT   -  INPUT statement execution
1977            **
1978            ** Category:   STEXEC
1979            **
1980            ** Purpose:
1981            **      Implements INPUT statement
1982            **
1983            ** Entry:
1984            **      (PCADDR) points to statement length
1985            **
1986            ** Exit:
1987            **      Exits through NXTSTM
1988            **
1989            ** Algorithm:
1990            **    INPUT:
1991            **      If prompt not specified then use default prompt
1992            **      Else if default input not specified then uses null
1993            **        string for default input
1994            **      Send prompt to display followed by default input and
1995            **        position cursor to start of default input.
1996            **    INP045:
1997            **      Initialize command stack pointer
1998            **      If ATTN key has been pressed then
1999            **        go to INPSTP
2000            **      Collapse math stack
2001            **      Call Character Editor (CHEDIT)
2002            **      If immediate execute key terminated CHEDIT then
2003            **         go to IEXKEY
2004            **      Jump to handle terminating key
2005            **         ATTN   --> INPATN
2006            **         CONT   --> INPEND
2007            **         UP     --> INPUP
2008            **         DOWN   --> INPDWN
2009            **         TOP    --> INPTOP
2010            **         BOTTOM --> INPTOP
2011            **         CMDS   --> INPBOT
2012            **         OFF    --> INPOFF
2013            **         otherwise ...
2014            **      Send cursor far right, CR/LF to display
2015            **      Edit input line into command stack
2016            **      Check if enough available memory
2017            **      Move input text onto math stack
2018            **      Initialize buffer pointer and length fields on stack
2019            **      Update (AVMEM) to protect this information on stack
2020            **      Skip prompt and default input (if any) and point to
2021            **        variable list in statement
2022            **      If this is really a LINPUT statement then
2023            **        jump to continue processing that statement
2024            **      Save start of variable name pointer for TRACE
2025            **      Evaluate destination expression
2026            **      Determine if destination is string or numeric
```

```
2027            **      Store destination address
2028            **      Save program counter
2029            **      Pop destination item off stack
2030            **      Point to buffer
2031            **      Call expression parser
2032            **      If numeric destination then
2033            **        goto NINP10
2034            **    SINP10:
2035            **      If expression parsed was numeric then
2036            **        goto Un"dSt
2037            **      If expression wasn't followed by a comma or eol then
2038            **        goto Un"dSt
2039            **    SINP20:
2040            **      Compile an expression terminator byte
2041            **      Restore pointer to comma or EOL
2042            **      Update input stream pointer
2043            **      Move code buffer onto math stack
2044            **      Point expression PC and stack pointer to interface
2045            **         of code and available memory
2046            **      Call expression execute and store result in
2047            **        destination variable
2048            **      Pop value off math stack
2049            **      Skip down stack to find input string pointer
2050            **      Move buffer pointer past comma or CR
2051            **    SINP40:
2052            **      Read next program byte
2053            **      If this byte is a comma then
2054            **        if not just past comma then
2055            **          give error "Too few inputs"
2056            **          go to INP310
2057            **        else loop back for next variable
2058            **      else
2059            **        if not at end of input stream then
2060            **          give error "Too many inputs"
2061            **          go to INP310
2062            **        else done with statement
2063            **
2064            **    NINP10:
2065            **      If valid numeric expression found and
2066            **        it is followed by comma or CR then
2067            **        go to SINP20
2068            **      Give error "Numeric Input"
2069            **      Goto INP310
2070            **
2071            **    Un"dSt:
2072            **      Get limits of available memory
2073            **      Calculate current position in buffer
2074            **      Skip leading blanks in unquoted string
2075            **      Output characters until comma or CR found
2076            **      Erase any trailing blanks
2077            **      Append a string header to string on stack
2078            **      Store string into variable
2079            **      Pop string off stack
2080            **      Go to SINP40
2081            **
```

```
2082                **    INP310:
2083                **      Display error message and reprompt
2084                **      go to INP045
2085                **
2086                ** History:
2087                **
2088                **    Date     Programmer          Modification
2089                **   --------  ----------  ---------------------------------
2090                **   10/18/83  B.S.        Updated Documentation
2091                **
2092                ********************************************************************
2093                ********************************************************************
2094 18699 00000 0          REL(5) =INPTDC       Decompile address
2095 1869E 00000 0          REL(5) =INPUTP       Parse address
2096 186A3         =INPUT
2097 186A3 7974             GOSUB  STMTST        Point to start of prompt(if any)
2098 186A7 133              AD1EX
2099 186AA 130              D0=A                 Move pointer to D0
2100 186AD 103              R3=A                 Save possible prompt start
2101 186B0 5F3              GONC   INP020
2102 186B3 161    INP010    D0=D0+ 2
2103 186B6 14A              A=DAT0 B             Read next byte
2104 186B9 966              ?A#C   B             Is it closing quote?
2105 186BC 7F               GOYES  INP010        No, then loop back
2106 186BE 161              D0=D0+ 2             Skip closing quote
2107 186C1 14A              A=DAT0 B             Read next byte
2108 186C4 3100             LC(2)  =tSEMIC
2109 186C8 962              ?A=C   B             Is it a semicolon?
2110 186CB 92               GOYES  INP030        Yes, then use null default input
2111 186CD 775E             GOSUB  EXPEXj        Evaluate default input expression
2112 186D1 8F00 000         GOSBVL =REVPOP       Call REV$ and POP1S
2113 186D8 1C2              D1=D1- 3
2114 186DB E4               A=A+1  A
2115 186DD E4               A=A+1  A             Add 2 to length to simulate
2116                                               CR on end
2117 186DF 1513             DAT1=A X             Create a pseudo buffer
2118 186E3 172              D1=D1+ 3
2119 186E6 137              CD1EX                Copy buffer pointer to C
2120 186E9 7213             GOSUB  GPRMPT        Get prompt
2121 186ED 4D0              GOC    INP040        (B.E.T.)  Save buffer ptr in INBS
2122                *-
2123                *-
2124 186F0 7E13   INP020    GOSUB  GDEFPR        Get default prompt
2125 186F4 3402   INP030    LC(5)  =ZERBUF
           B81
2126 186FB 7F13   INP040    GOSUB  REPROM        Reprompt for input
2127 186FF 8F00   INP045    GOSBVL =CMD1ST       Initialize command stack pointer
           000
2128 18706 8F00   INP050    GOSBVL =CK"ON"       Has ATTN been pressed?
           000
2129 1870D 460              GOC    INP070        No, then okay
2130 18710 6BE3             GOTO   INPSTP        Yes, then halt statement
```

```
2131                *-
2132                *-
2133 18714 7538   INP070 GOSUB  collap          Collapse math stack
2134 18718 8E00          GOSUBL =CHEDIT         Let user input line
           00
2135 1871E 460           GOC    INP080          Skip if not IEX key
2136 18721 6F34          GOTO   INPIEX
2137                *-
2138                *-
2139 18725         INP080
2140 18725 8E00          GOSUBL =FINDAj
           00
2141 1872B 00            CON(2) =kcATTN         Attention
2142 1872D F83           REL(3) INPATN
2143 18730 00            CON(2) =kcCONT         CONTinue key
2144 18732 BD3           REL(3) INPEND
2145 18735 00            CON(2) =kcUP           Cursor up key
2146 18737 C53           REL(3) INPUP
2147 1873A 00            CON(2) =kcDOWN         Cursor down key
2148 1873C D63           REL(3) INPDWN
2149 1873F 00            CON(2) =kcTOP          Cursor top key
2150 18741 273           REL(3) INPTOP
2151 18744 00            CON(2) =kcBOT          Cursor bottom key
2152 18746 923           REL(3) INPBOT
2153 18749 00            CON(2) =kcLAST         Command stack key
2154 1874B 423           REL(3) INPBOT          (Same as cursor bottom)
2155 1874E 00            CON(2) =kcOFF          Off key (or 10 minute timeout)
2156 18750 9F3           REL(3) INPOFF
2157 18753 00            CON(2) 0               Otherwise
2158 18755 71E2          GOSUB  FINLIN
2159 18759 8F00          GOSBVL =MAKEBF
           000
2160                * D0 Points past text in buffer.  A(A)=Len+3
2161
2162 18760 D2            C=0    A
2163 18762 303           LC(1)  3               Need 3 more nibbles
2164 18765 7EAD          GOSUB  CHKMEM
2165 18769 D6            C=A    A               C(A)=Length to move
2166 1876B 8E76          GOSUBL moved3          Copy buffer onto stack
           40
2167                                              (including length)
2168 18771 3260  INP150  LC(3)  6               Point to first byte of buffer
           0
2169 18776 1553          DAT1=C X               Current buffer position is 6
2170 1877A 7A34          GOSUB  AVE=D1          Update AVMEME to D1 pointer
2171 1877E 7E93          GOSUB  STMTST          Check for a user supplied prompt
2172 18782 5B0           GONC   INP200          Not found then don't skip it
2173 18785 8E00          GOSUBL =EXPSKP         Skip expression
           00
2174 1878B 171           D1=D1+ 2               Skip semicolon
2175 1878E         INP200
2176
2177 1878E 3100          LC(2)  =tLINPT
2178 18792 965           ?B#C   B
2179 18795 60            GOYES  INP205
```

```
2180 18797 6454          GOTO   LINPT+
2181              *-
2182              *-
2183 1879B         INP205
2184 1879B 137           CD1EX
2185 1879E 8F00  INP210  GOSBVL =SETTRC        Save start of dest for TRACE
           000
2186 187A5 7F7D          GOSUB  EXPEXj         Execute expression
2187              *   Now test if string or numeric
2188 187A9 AC9           C=B    S
2189 187AC 855           ST=1   S/NFlg
2190 187AF 80DF          P=C    15
2191 187B3 89C           ?P=    #C             Is it a C?
2192 187B6 A0            GOYES  S/N?10
2193 187B8 89D           ?P=    #D             Is it a D?
2194 187BB 50            GOYES  S/N?10
2195 187BD 845           ST=0   S/NFlg
2196 187C0 20    S/N?10  P=     0
2197 187C2 7C74          GOSUB  DEST+          Store desination address
2198 187C6 136           CDOEX
2199 187C9 1B00          DO=(5) =STMTDO
           000
2200 187D0 144           DATO=C A              Save program PC
2201 187D3 775D          GOSUB  popnth         Get rid of item
2202 187D7 7412          GOSUB  GETPOS         Point A(A) into buffer, set carry
2203 187DB 7900          GOSUB  EXCPAR
2204 187DF 875           ?ST=1  S/NFlg         Is it string?
2205 187E2 03            GOYES  SINP10         Yes, then parse as string
2206 187E4 6AE0          GOTO   NINP10         No, then parse as numeric
2207              *-
2208              *-
```

```
2209                    EJECT
2210                ****************************************************************
2211                ****************************************************************
2212                **
2213                ** Name:(S) EXCPAR  -  Execution Time Expression Parse
2214                **
2215                ** Category:   EXCUTL
2216                **
2217                ** Purpose:
2218                **      Parses an expression in the constraints of an
2219                **      executing statement.
2220                **
2221                ** Entry:
2222                **      Carry clear:  D1 contains pointer to input stream
2223                **      Carry set:  A(A) contains pointer to input stream
2224                **      The pointer to the input stream is also used as
2225                **        a starting point for the parse stack.
2226                **      (AVMEMS) is start of output buffer
2227                **      P      =  0
2228                **
2229                ** Exit:
2230                **      P      =  0
2231                **      (AVMEME) = D1 on entry
2232                **      See exit conditions for EXPPAR
2233                **
2234                ** Calls:      AVE=D1,EXPP10
2235                **
2236                ** Uses.......
2237                **  Exclusive: C,D0,D1,R3,(AVMEME)
2238                **  Inclusive: A,B,C,D,D0,D1,R0,R1,R3
2239                **
2240                ** Stk lvls:   3
2241                **
2242                ** History:
2243                **
2244                **    Date      Programmer              Modification
2245                **  --------   ----------   ---------------------------------
2246                **  11/01/83   B.S.         Added documentation
2247                **
2248                ****************************************************************
2249                ****************************************************************
2250 187E8 1B00 =EXCPAR D0=(5) =AVMEMS
          000
2251 187EF 146          C=DAT0 A              Get start of buffer pointer
2252 187F2 134          D0=C                  Init output pointer
2253 187F5 10B          R3=C                  Save (AVMEMS) in R3
2254 187F8 7CB3         GOSUB  AVE=D1         D1,AVMEME=(D1)
2255 187FC D7           D=C    A              D(A)=Expr parse stack pointer
2256 187FE 550          GONC   EXCP10         Parse at A(A) or D1
2257 18801 131          D1=A                  Start parsing at A(A)
2258 18804 847   EXCP10 ST=0   7              Not parsing left hand side of "="
2259 18807 8D00         GOVLNG =EXPP10        Parse expression
          000
2260                *-
2261                *-
```

```
2262 1880E 6B51  Un"dSt GOTO   UN"DST
2263            *-
2264            *-
2265 18812 873   SINP10 ?ST=1  NumExp          Valid String expression found?
2266 18815 9F           GOYES  Un"dSt          No, then parse as unquoted string
2267 18817 3100         LC(2)  =tCOMMA
2268 1881B 962          ?A=C   B               Is a comma next?
2269 1881E A0           GOYES  SINP20          Yes, then okay
2270 18820 300          LC(1)  =tEOL
2271 18823 966          ?A#C   B               Is it at EOL?
2272 18826 8E           GOYES  Un"dSt          No, then parse as unquoted string
2273 18828 8F00  SINP20 GOSBVL =OUT1TK         Output terminator byte
          000
2274 1882F 8F00         GOSBVL =RESPTR         Restore pointer to "," or EOL
          000
2275            *                              A(A)=RESPTR value
2276 18836 771E         GOSUB  D1=AVE
2277 1883A EA           A=A-C  A               Calculate updated pointer
2278 1883C 1513         DAT1=A X               Write out updated pointer
2279 18840 7683         GOSUB  AVMMOV          Move buffer onto stack
2280 18844 137          CD1EX
2281 18847 134          D0=C                   Code starts here too
2282 1884A 135          D1=C                   Start stack pointer here
2283 1884D 1C4          D1=D1- 5               Start stack pointer here
2284 18850 111          A=R1                   Recall code length
2285 18853 141          DAT1=A A               Length of code to pop off later
2286            *       The following "fancy GOSUB" is needed to keep the PC in
2287            *       STMTD0 properly maintained throughout possible user
2288            *       defined function execution
2289 18856 34C6         LC(5)  =SINP30         Return to SINP30 after...
          881
2290 1885D 06   =EX-STC RSTK=C
2291 1885F 3400         LC(5)  =STORE
          000
2292 18866 06           RSTK=C                 ...returning to STORE after...
2293 18868 6279  EXPRj1 GOTO   EXPRj           ...doing EXPR (evaluate expr)
2294            *-
2295            *-
2296 1886C 7EBC =SINP30 GOSUB  popnth          Pop value off stack
2297 18870 143          A=DAT1 A
2298 18873 137          CD1EX
2299 18876 C2           C=C+A  A
2300 18878 134          D0=C
2301 1887B 164          D0=D0+ 5               Skip D0 input buffer
2302 1887E D0           A=0    A
2303 18880 1523         A=DAT0 X               Read input pointer
2304 18884 E4           A=A+1  A
2305 18886 E4           A=A+1  A
2306 18888 1503         DAT0=A X               Move buffer ptr past comma or CR
2307 1888C 136          CD0EX
2308 1888F CA           A=A+C  A               Calculate current position
2309 18891 1B00  SINP40 D0=(5) =AVMEME
          000
2310 18898 144          DAT0=C A               Set AVMEME back to buffer pointer
2311 1889B 131          D1=A                   D1 points into buffer
```

```
2312 1889E 1C1          D1=D1- 2              Move D1 back to comma or CR
2313 188A1 1A00         D0=(4) =STMTD0
           00
2314 188A7 146          C=DAT0 A              Read program PC
2315 188AA 134          D0=C                  D0= program PC
2316 188AD 14A          A=DAT0 B              Read in program byte
2317 188B0 161          D0=D0+ 2              Skip past program byte
2318 188B3 3100         LC(2)  =tCOMMA        Comma token
2319 188B7 966          ?A#C   B              Is there a comma token next?
2320 188BA 35           GOYES  INP300         No, then must be at end
2321             * Now need to find a comma in input line
2322 188BC 14B          A=DAT1 B
2323 188BF 31C2         LCASC  \,\
2324 188C3 966          ?A#C   B              Comma next char?
2325 188C6 E2           GOYES  ER:COM         No, then error
2326 188C8 136          CD0EX                 Move program pointer to C(A)
2327 188CB 62DE         GOTO   INP210         Loop back for next variable
2328             *-
2329             *-
2330 188CF 870   NINP10 ?ST=1  InvalE         Valid expression found?
2331 188D2 B2           GOYES  ER:NUM         No, then error
2332 188D4 863          ?ST=0  NumExp         Numeric expression found?
2333 188D7 62           GOYES  ER:NUM         No, then error
2334 188D9 3100         LC(2)  =tCOMMA        Comma token
2335 188DD 962          ?A=C   B              Is it a comma token?
2336 188E0 A0           GOYES  NINP20         Yes, then okay
2337 188E2 300          LC(1)  =tEOL          EOL token
2338 188E5 966          ?A#C   B              Is it an EOL token?
2339 188E8 51           GOYES  ER:NUM         No, then error
2340 188EA 6D3F  NINP20 GOTO   SINP20        Fall into string input code
2341             *-
2342             *-
2343 188EE 8C00 =inexit GOLONG =NXtstm
           00
2344             *-
2345             *-
2346 188F4 3300  ER:COM LC(4)  =eTOOFI        Too Few Inputs error
           00
2347 188FA 442          GOC    INP310         (B.E.T.)
2348             *-
2349             *-
2350 188FD 8F00  ER:NUM GOSBVL =RESPTR        Back up input pointer
           000
2351 18904 3300         LC(4)  =eNUMIN        Numeric input error
           00
2352 1890A 541          GONC   INP310         (B.E.T.)
2353
2354             *-
2355             *-
2356 1890D 14B   INP300 A=DAT1 B              Read next char in input line
2357 18910 31D0         LCHEX  0D             Carriage return
2358 18914 962          ?A=C   B              Is it a carriage return?
2359 18917 7D           GOYES  inexit         Yes, done with INPUT statement
2360 18919 3300         LC(4)  =eTOOMI
           00
```

```
2361 1891F 109    INP310 R1=C                Save error num
2362 18922 1B00          DO=(5) =AVMEME
           000
2363 18929 142           A=DATO A
2364 1892C 133           AD1EX
2365 1892F 175           D1=D1+ 6             Advance ptr to start of buffer
2366 18932 102           R2=A                 Save pointer into buffer
2367 18935 133           AD1EX
2368 18938 1A00          DO=(4) =INBS
           00
2369 1893E 140           DAT0=A A             Point INBS to start of buffer
2370 18941 7BD1          GOSUB  STMTST        Check for user supplied prompt
2371 18945 137           CD1EX                C(A)=Prompt string address
2372 18948 490           GOC    INP320        Is this a prompt?  Yes, then okay
2373 1894B 3451          LC(5)  =DEFPRM       No, then use default prompt
           B81
2374 18952 DA     INP320 A=C    A             Copy prompt address to A(A)
2375 18954 11A           C=R2
2376 18957 135           D1=C                 D1 points into buffer
2377 1895A 119           C=R1                 C(A)=Error number
2378 1895D 2A            P=     10            Parse error
2379                                          Store ERRN
2380                                          No ERRN disp
2381                                          ON ERROR
2382 1895F 8F00          GOSBVL =MFERR*       Reprompt for input
           000
2383 18966 689D          GOTO   INP045
2384            *-
2385            *-
2386 1896A 7FDC  UN"DST  GOSUB  GETAVM        D(A)=(AVMEMS)   (for STKCHR)
2387                                          D1=(AVMEME)
2388 1896E 7D70          GOSUB  GETPOS        A(A)=Ptr into buf, C(A)=(AVMEME)
2389 18972 131           D1=A                 DO points into buffer
2390 18975 134           DO=C
2391 18978 109           R1=C                 Start of output string
2392                                            (for ADHEAD)
2393 1897B 8F00          GOSBVL =GNXTCR       Skip leading blanks
           000
2394 18982 136           CDOEX
2395 18985 137           CD1EX
2396 18988 136           CDOEX
2397 1898B 137   UN"D10  CD1EX
2398 1898E D5            B=C    A             Save address of last output byte
2399 18990 135           D1=C
2400 18993 14A   UN"D20  A=DAT0 B             Read next byte
2401 18996 161           DO=DO+ 2             Skip past byte
2402 18999 31C2          LCASC  \,\
2403 1899D 962           ?A=C   B             Is it a comma?
2404 189A0 D1            GOYES  UN"D30        Yes, then exit loop
2405 189A2 31D0          LCHEX  0D
2406 189A6 962           ?A=C   B             Is it a CR?
2407 189A9 41            GOYES  UN"D30        Yes, then exit loop
2408 189AB D6            C=A    A             C(B)=Char to add to stack
2409 189AD 735B          GOSUB  STKCHR        Add char to stack
2410 189B1 3102          LCASC  \ \
```

```
2411 189B5 966          ?A#C   B              Is it a blank?
2412 189B8 3D           GOYES  UN"D10         No, then remember address
2413 189BA 58D          GONC   UN"D20         (B.E.T.) Yes, then just continue
2414            *-
2415            *-
2416 189BD 136   UN"D30 CDOEX                 C(A)=Points past comma or CR
2417 189C0 1B00         DO=(5) =AVMEME
           000
2418 189C7 142          A=DATO A
2419 189CA 130          DO=A   A
2420 189CD E2           C=C-A  A
2421 189CF 1543         DATO=C X               Update buffer pointer
2422 189D3 D9           C=B    A               Recall last non-blank char addr
2423 189D5 135          D1=C                   D1=last nonblank
2424 189D8 850          ST=1   Return          Return from ADHEAD
2425 189DB 7DAC         GOSUB  adhead          Append string header
2426 189DF 76C1         GOSUB  D1STOR          Store this string into variable
2427 189E3 774B         GOSUB  popnth          Pop string off stack
2428 189E7 7400         GOSUB  GETPOS          Set A(A) to point into buffer
2429 189EB 65AE         GOTO   SINP40          Pretend string exprress has been
2430                                             executed
2431            *-
2432            *-
2433            * RETURNS WITH A(A)=Pointer into buffer.  C(A),D1=(AVMEME)
2434 189EF D0    GETPOS A=0    A
2435 189F1 1533         A=DAT1 X
2436 189F5 137          CD1EX
2437 189F8 CA           A=A+C  A
2438 189FA 135          D1=C
2439 189FD 02           RTNSC
2440            *-
2441            *-
2442 189FF 10B   GPRMPT R3=C                   R3=buffer pointer
2443 18A02 7A11         GOSUB  STMTST
2444 18A06 137          CD1EX                  C(A)=Prompt pointer
2445                                             (if supplied by user)
2446 18A09 12B          CR3EX                  C(A)=buffer pointer
2447                                           R3=Prompt pointer
2448 18A0C 400          RTNC                   Return if user supplied prompt
2449 18A0F 12B          CR3EX
2450 18A12 3451  GDEFPR LC(5)  =DEFPRM
           B81
2451 18A19 12B          CR3EX
2452 18A1C 01           RTN
2453            *-
2454            *-
2455            ****************************************************************
2456            ****************************************************************
2457            **
2458            ** Name:(S) REPROM  -  Reprompt for input
2459            **
2460            ** Category:   EXCUTL
2461            **
2462            ** Purpose:
2463            **      Sends buffer to display following prompt and positions
```

```
2464            **      cursor to start of line.
2465            **
2466            ** Entry:
2467            **      C(A) = Pointer to buffer to be displayed
2468            **      R3(A) = Pointer to quoted string that is prompt
2469            **
2470            ** Exit:
2471            **      Exits via DONNA
2472            **
2473            ** Calls:      DONNA
2474            **
2475            ** Uses.......
2476            **  Inclusive: A,B,C,D,D0,D1,R3
2477            **
2478            ** Stk lvls:   4
2479            **
2480            ** History:
2481            **
2482            **    Date      Programmer              Modification
2483            **  --------    ----------   ------------------------------------
2484            **  11/01/83    B.S.         Added documentation
2485            **
2486            ********************************************************************
2487            ********************************************************************
2488 18A1E 1F00 =REPROM D1=(5) =INBS
           000
2489 18A25 145          DAT1=C A              Save buffer pointer in INBS
2490 18A28 11B          C=R3
2491 18A2B AF0          A=0    W
2492 18A2E DA           A=C    A
2493 18A30 103          R3=A                  Position cursor at begin of
2494                                            default input
2495            *                          Note: R3(S) must=0 for call to DONNA!
2496 18A33 8D00         GOVLNG =DONNA         Send prompt and default input to
           000
2497                                            display
2498            *-
```

```
2499                    EJECT
2500                ****************************************************************
2501                ****************************************************************
2502                **
2503                ** Name:(S) FINLIN  -  Finish line in display/video
2504                **
2505                ** Category:   DSPUTL
2506                **
2507                ** Purpose:
2508                **      Finishes line in display and video by moving the
2509                **      cursor to the far right then sending CR/LF with
2510                **      no delay.
2511                **
2512                ** Entry:
2513                **      P       =  0
2514                **
2515                ** Exit:
2516                **      Carry clear
2517                **      P       =  0
2518                **
2519                ** Calls:      CURSFR,CRLFOF
2520                **
2521                ** Uses.......
2522                **  Inclusive: A(W),B(W),C(W),D(W),D0,D1,ST(11-0)
2523                **
2524                ** Stk lvls:   4
2525                **
2526                ** Algorithm:
2527                **      Unprotect last display buffer character
2528                **         (This is needed to guarantee that even if the entire
2529                **           display line is protected the cursor can be moved
2530                **           past the last character on the video monitor line
2531                **           which will allow a CR/LF sent to the monitor to
2532                **           position the cursor past the last video line of
2533                **           this display line.)
2534                **      Send cursor far right.
2535                **      Restore protection bit of last character.
2536                **      Send replace cursor, CR/LF with no delay.
2537                **      Return with carry clear.
2538                **
2539                ** History:
2540                **
2541                **    Date      Programmer              Modification
2542                **  --------   ----------   ---------------------------------
2543                **  11/01/83   B.S.         Added documentation
2544                **
2545                ****************************************************************
2546                ****************************************************************
2547 18A3A 1B00 =FINLIN D0=(5) =DSPMSK
           000
2548 18A41 14E          C=DAT0 B
2549 18A44 0A           ST=C
2550 18A46 840          ST=0   0                Unprotect last display buf char
2551 18A49 0B           CSTEX
2552 18A4B 14C          DAT0=C B
```

```
2553 18A4E 8E00          GOSUBL =CURSFR
           00
2554 18A54 1A00          DO=(4) =DSPMSK
           00
2555 18A5A 09            C=ST
2556 18A5C 14C           DATO=C B
2557 18A5F 8F00          GOSBVL =CRLFOF        Cursor off,Rep cur,CRLF no delay
           000
2558 18A66 03            RTNCC
2559              *-
2560              *-
2561 18A68 8D00  cmdini  GOVLNG =CMDINI
           000
2562              *-
2563              *-
2564 18A6F 8F00  INPBOT  GOSBVL =CMD1ST
           000
2565 18A76 70CF  INPB10  GOSUB  FINLIN         Finish current line
2566 18A7A 8F00          GOSBVL =CMDFND        Find buffer
           000
2567 18A81 172           D1=D1+ 3
2568 18A84 137           CD1EX                 C points to start of buffer
2569
2570 18A87 747F          GOSUB  GPRMPT         Get prompt
2571 18A8B 7F8F          GOSUB  REPROM         Reprompt for input
2572 18A8F 667C          GOTO   INPO50         Redo INPUT statement
2573              *-
2574              *-
2575 18A93 71DF  INPUP   GOSUB  cmdini         Read current and max command
2576 18A97 B04           A=A+1  P              Increment command number
2577 18A9A 4BD           GOC    INPB10         Watch for overflow (do a NOP)
2578 18A9D 986   INPU10  ?A>C   P              Greater than max command?
2579 18AA0 6D            GOYES  INPB10         Yes, then don't change it
2580 18AA2 1590  INPU20  DAT1=A 1              Change command number
2581 18AA6 5FC           GONC   INPB10         (B.E.T.)
2582              *-
2583              *-
2584 18AA9 7BBF  INPDWN  GOSUB  cmdini         Read current and max command
2585 18AAD CC            A=A-1  A
2586 18AAF 6DEF          GOTO   INPU10
2587              *-
2588              *-
2589 18AB3 71BF  INPTOP  GOSUB  cmdini         Read current and max command
2590 18AB7 DA            A=C    A              Set command number to max
2591 18AB9 58E           GONC   INPU20         (B.E.T.)
2592              *-
2593              *-
2594 18ABC 7191  INPATN  GOSUB  atnclr         Clear attention flag
2595 18AC0 7D8B          GOSUB  D1=AVE
2596 18AC4 850           ST=1   Return         Return from DSP$00
2597 18AC7 701B          GOSUB  DSP$00
2598 18ACB 8F00          GOSBVL =POP1S
           000
2599 18AD2 3102          LC(2)  \ \
2600 18AD6 D5            B=C    A
```

```
2601 18AD8 CC    INPA10 A=A-1  A
2602 18ADA CC           A=A-1  A              Subtract 2(CR is ignored)
2603 18ADC 8A8          ?A=0   A              Is it a null buffer?
2604 18ADF D1           GOYES  INPSTP         Yes, then abort INPUT statement
2605 18AE1 171          D1=D1+ 2
2606 18AE4 14F          C=DAT1 B
2607 18AE7 961          ?B=C   B
2608 18AEA EE           GOYES  INPA10
2609 18AEC 8E00         GOSUBL =CURSFL        Move cursor to far left
           00
2610 18AF2 8E00         GOSUBL =-LINE         Clear remainder of line
           00
2611 18AF8 6B1C         GOTO   INP070         Redo INPUT statement
2612            *-
2613            *-
2614 18AFC 7A3F  INPSTP GOSUB  FINLIN
2615 18B00 8E00         GOSUBL =NOSCRL
           00
2616 18B06 85E          ST=1   =NoCont
2617 18B09 6A40         GOTO   INP010
2618            *-
2619            *-
2620 18B0D 792F =INPEND GOSUB  FINLIN
2621 18B11 6CDD         GOTO   inexit
2622            *-
2623            *-
2624 18B15 22F3 =DEFPRM NIBASC \"? "\         Default prompt
           0222
2625            *-
2626            ******************************************************************
2627            ******************************************************************
2628            **
2629            ** Name:(S) ZERBUF  -  Looks Like a Zero Length Buffer
2630            **
2631            ** Category:   STEXEC
2632            **
2633            ** Purpose:
2634            **      This looks like a zero length buffer.
2635            **
2636            ** Entry:
2637            **      Do not enter
2638            **
2639            ** History:
2640            **
2641            **    Date     Programmer              Modification
2642            **  --------   ----------   ------------------------------------
2643            **  11/09/83   B.S.         Added documentation
2644            **
2645            ******************************************************************
2646            ******************************************************************
2647            *-
2648 18B1D 000          CON(3) 0
2649 18B20      =ZERBUF
2650            *-
2651            *-
```

```
2652                    $TITLE STMTST - Get start of statement
2653                ********************************************************************
2654                ********************************************************************
2655                **
2656                ** Name:    STMTST  -  Get address of start of statement
2657                **
2658                ** Category:   EXCUTL
2659                **
2660                ** Purpose:
2661                **      Calculates address of start of statement by looking at
2662                **      PCADDR.
2663                **
2664                ** Entry:
2665                **      P      =  0
2666                **
2667                ** Exit:
2668                **      P      =  0
2669                **      B(B)   =  Statement token
2670                **      A(B)   =  Byte following statement token
2671                **      D1 points past statement token or past zero byte
2672                **         if it follows statement token
2673                **      Carry set iff byte following statement token is zero
2674                **
2675                ** Calls:      D1=@D1
2676                **
2677                ** Uses.......
2678                **  Inclusive: A(B),B(A),C(A),D1
2679                **
2680                ** Stk lvls:   1
2681                **
2682                ** History:
2683                **
2684                **    Date     Programmer            Modification
2685                **  --------   ----------   ------------------------------------
2686                **  10/18/83   B.S.         Added Documentation
2687                **
2688                ********************************************************************
2689                ********************************************************************
2690 18B20 1F00   =STMTST D1=(5) =PCADDR
           000
2691 18B27 7D2B          GOSUB  D1=@D1
2692 18B2B 171           D1=D1+ 2              Skip statement length
2693 18B2E 14B           A=DAT1 B              Read statement token
2694 18B31 D8            B=A    A              Save statement token in B
2695 18B33 171           D1=D1+ 2              Point at statement start
2696 18B36 14B           A=DAT1 B              Read possible zero byte
2697 18B39 171           D1=D1+ 2              Point at possible opening quote
2698 18B3C 14F           C=DAT1 B              Read possible opening quote
2699 18B3F 968           ?A=0   B              Is there a user-prompt?
2700 18B42 00            RTNYES                Yes, then return with carry set
2701 18B44 1C1           D1=D1- 2              No, then move pointer back
2702 18B47 03            RTNCC                 Return with carry set
2703                *-
```

```
2704                    EJECT
2705                ****************************************************************
2706                ****************************************************************
2707                **
2708                ** Name:(S) INPOFF  -  Restart statement after DSLEEP
2709                **
2710                ** Category:   EXCUTL
2711                **
2712                ** Purpose:
2713                **      Allows a statement to set itself to be restarted
2714                **      if continue is pressed, then turns off machine.
2715                **      ATTN key will send machine back to BASIC interpreter
2716                **      which will suspend execution.
2717                **
2718                ** Entry:
2719                **      P       =  0
2720                **
2721                ** Exit:
2722                **      Exits through MFERRS
2723                **
2724                ** Calls:      FINLIN,DSLEEP,MFER42,MFERRS
2725                **
2726                ** Stk lvls:   6
2727                **
2728                ** History:
2729                **
2730                **    Date      Programmer              Modification
2731                **  --------    ----------   ------------------------------------
2732                **  11/01/83    B.S.         Added documentation
2733                **
2734                ****************************************************************
2735                ****************************************************************
2736 18B49 7DEE  =INPOFF GOSUB  FINLIN          Finish display line
2737 18B4D 8F00          GOSBVL =DSLEEP        Go to sleep
           000
2738 18B54 8F00  =INP010 GOSBVL =MFER42        Reset CONT information
           000
2739 18B5B 8C00          GOLONG =mferrs
           00
2740                ****
```

```
2741                    EJECT
2742             *  A(A)=R3(3)=Definition length + 1 nibbles
2743             *  D1 points at first character of definition
2744 18B61 137    INPIEX CD1EX
2745 18B64 109           R1=C                  Save pointer to first char of def
2746 18B67 7FCE          GOSUB  FINLIN         Finish line
2747 18B6B 113           A=R3                  Recall def len + 1
2748 18B6E D2            C=0    A
2749 18B70 307           LC(1)  7              Need 7 more nibbles
2750             *                             Have 1,need 8 total (6+2 for CR)
2751 18B73 70A9          GOSUB  CHKMEM
2752 18B77 135           D1=C
2753 18B7A 129           CR1EX                 R1 saves buffer ptr ptr
2754                                           C=Start of def
2755 18B7D 134           D0=C                  D0 points to start of def
2756 18B80 172           D1=D1+ 3              Point D1 at length field (3 nibs)
2757 18B83 11B           C=R3
2758 18B86 E6            C=C+1  A              Calculate length counting CR
2759 18B88 15D2          DAT1=C 3              Write out length
2760 18B8C 172           D1=D1+ 3              D1=Start of destination
2761 18B8F 8E00          GOSUBL =MOVEU3        Move key def onto stack
          00
2762 18B95 1C1           D1=D1- 2              Point where CR needs to go
2763 18B98 31D0          LCHEX  0D
2764 18B9C 14D           DAT1=C B              Write out CR
2765 18B9F 119           C=R1                  Recall where buffer ptr goes
2766 18BA2 135           D1=C                  D1 points to buffer ptr
2767 18BA5 6BCB          GOTO   INP150         Rejoin normal code
```

```
2768                    STITLE D1STOR
2769                ****************************************************************
2770                ****************************************************************
2771                **
2772                ** Name:    D1STOR  -  Update AVMEME and STORE into variable
2773                **
2774                ** Category:   EXCUTL
2775                **
2776                ** Purpose:
2777                **      Updates (AVMEME) to D1, reads A(W) from top of stack
2778                **      and calls STORE to store into variable determined
2779                **      previously by DEST routine.
2780                **
2781                ** Entry:
2782                **      D1 points to top of math stack
2783                **      Statement scratch contains info saved by DEST
2784                **
2785                ** Exit:
2786                **      P      =  0
2787                **
2788                ** Calls:      AVE=D1,STORE
2789                **
2790                ** Uses.......
2791                **  Inclusive: Same as STORE
2792                **
2793                ** Stk lvls:   Same as STORE
2794                **
2795                ** History:
2796                **
2797                **    Date     Programmer              Modification
2798                **  --------   ----------   ----------------------------------
2799                **  10/18/83   B.S.         Updated documentation
2800                **
2801                ****************************************************************
2802                ****************************************************************
2803 18BA9 7B00 =D1STOR GOSUB  AVE=D1
2804 18BAD 1537         A=DAT1 W              Read top of stack
2805 18BB1 8D00  store  GOVLNG =STORE
           000
2806                *-
2807                *-
```

```
2808                    STITLE AVE=D1 - Set stack pointer
2809                ****************************************************************
2810                ****************************************************************
2811                **
2812                ** Name:(S) AVE=D1  -  Update AVMEME From D1 or C
2813                ** Name:(S) AVE=C   -  Update AVMEME From D1 or C
2814                **
2815                ** Category:   PTRUTL
2816                **
2817                ** Purpose:
2818                **      Update AVMEME pointer to the value in D1 or C
2819                **
2820                ** Entry:
2821                **      AVE=D1 :  D1 = new value for AVMEME
2822                **      AVE=C  :  C(A) = new value for AVMEME
2823                **
2824                ** Exit:
2825                **      C(A)=D1= Value stored into AVMEME
2826                **
2827                ** Calls:      None
2828                **
2829                ** Uses.......
2830                **  Inclusive: C(A)
2831                **
2832                ** Stk lvls:   0
2833                **
2834                ** History:
2835                **
2836                **    Date     Programmer             Modification
2837                **  --------   ----------   ---------------------------------
2838                **  10/12/82   B.S.         Added documentation
2839                **
2840                ****************************************************************
2841                ****************************************************************
2842 18BB8 137   =AVE=D1 CD1EX
2843 18BBB 1F00  =AVE=C  D1=(5) =AVMEME
           000
2844 18BC2 145           DAT1=C A
2845 18BC5 135           D1=C
2846 18BC8 01            RTN                    Leave carry unchanged
2847                *-
2848                *-
```

```
2849                    STITLE AVMMOV - Move buffer to stack
2850                ****************************************************************
2851                ****************************************************************
2852                **
2853                ** Name:    AVMMOV  -  Move Memory From AVMEMS To Before D1
2854                **
2855                ** Category:   MTHSTK
2856                **
2857                ** Purpose:
2858                **      Moves memory between AVMEMS and D0 to before D1,
2859                **      that is, onto the math stack.
2860                **
2861                ** Entry:
2862                **      D0=end of source
2863                **      D1=end of destination
2864                **      R3=(AVMEMS)
2865                **
2866                ** Exit:
2867                **              P=0
2868                **              R1=Length of buffer moved
2869                **
2870                ** Calls:      MOVED3,R1
2871                **
2872                ** Uses.......
2873                **  Inclusive:   See MOVED3
2874                **
2875                ** Stk lvls:   0
2876                **
2877                ** History:
2878                **
2879                **    Date      Programmer              Modification
2880                **  --------   ----------   ----------------------------------
2881                **  10/10/82   B.S.         Updated documentation
2882                **
2883                ****************************************************************
2884                ****************************************************************
2885 18BCA 132  =AVMMOV ADOEX                  A=End of output buffer
2886 18BCD 11B         C=R3
2887 18BD0 130         D0=A                    D0=end of source (output buffer)
2888 18BD3 EE          C=A-C  A                C(A)=Length of buffer
2889 18BD5 109         R1=C                    Save length
2890                *                          D1=End of destination
2891 18BD8 8C00  moved3 GOLONG =MOVED3         Move buffer to top of stack
           00
2892                *-
2893                *-
```

```
2894                    STITLE LINPUT - Statement execution
2895                ******************************************************************
2896                ******************************************************************
2897                **
2898                ** Name:    LINPUT  -  LINPUT statement execution
2899                **
2900                ** Category:   STEXEC
2901                **
2902                ** Purpose:
2903                **      Implements LINPUT statement
2904                **
2905                ** Entry:
2906                **      (PCADDR) points to line length before LINPUT statement
2907                **
2908                ** Exit:
2909                **      Exits through NXTSTM
2910                **
2911                ** Algorithm:
2912                **      Jumps to INPUT statement
2913                **
2914                **      INPUT statement jumps back to LINPT+ after prompting
2915                **        and getting response from user.
2916                **      Takes input string, adds a string header, reverses
2917                **        it to get normal stack item
2918                **      Evaluates destination address
2919                **      Stores input string into destination address
2920                **      Exits statement through NXTSTM
2921                **
2922                ** History:
2923                **
2924                **    Date     Programmer          Modification
2925                **  --------   ----------   ------------------------------------
2926                **  10/18/83   B.S.         Added documentation
2927                **
2928                ******************************************************************
2929                ******************************************************************
2930 18BDE 0000     0          REL(5) =INPTDC
2931 18BE3 0000     0          REL(5) =LINPTP
2932 18BE8 6ABA =LINPUT GOTO   INPUT
2933                *-
2934                *-
2935 18BEC 137   LINPT+ CD1EX
2936 18BEF 108         RO=C                    Save start of dest expr
2937 18BF2 775A        GOSUB  GETAVM           D1=AVE,D=AVMS
2938 18BF6 AF2         C=0    W                Clear C(6,5)
2939 18BF9 170         D1=D1+ 1                Point 2 nibs before 3 nib length
2940 18BFC 147         C=DAT1 A                Read buffer length
2941 18BFF 31F0        LCHEX  0F               String tag
2942 18C03 1CA         D1=D1- 11
2943 18C06 7DF8        GOSUB  STKCH+           Check for avail mem
2944 18C0A 1557        DAT1=C W                Write out complete header
2945 18C0E 8E00     0          GOSUBL =REV$             Reverse string
```

```
2946 18C14 8E00          GOSUBL =XXHEAD        Remove string header
           00
2947 18C1A 171           D1=D1+ 2              Throw away CR on end
2948 18C1D 850           ST=1   0              Return from ADHEAD
2949 18C20 786A          GOSUB  adhead         Put string header back on
2950 18C24 118           C=R0                  Recall start of dest expr
2951 18C27 8F00          GOSBVL =SETTRC        Set TRACE pointer
           000
2952 18C2E 763C          GOSUB  EXPRj1         Get destination address
2953 18C32 7C00          GOSUB  DEST+          Store destination
2954 18C36 74F8          GOSUB  popnth         Pop dest to expose string
2955 18C3A 7B6F          GOSUB  D1STOR         Store string
2956 18C3E 6FAC          GOTO   inexit         Now done with statement
```

```
2957                    EJECT
2958                ********************************************************************
2959                ********************************************************************
2960                **
2961                ** Name:    DEST+   -  Preserves D1 during DEST call
2962                **
2963                ** Category:   EXCUTL
2964                **
2965                ** Purpose:
2966                **      Calls DEST but preserves D1 by saving it in A(A)
2967                **
2968                ** Entry:
2969                **      Same as DEST
2970                **
2971                ** Exit:
2972                **      Same as DEST except D1 preserved
2973                **
2974                ** Calls:      DEST
2975                **
2976                ** Uses.......
2977                **  Inclusive: Same as DEST except A(A) destroyed
2978                **
2979                ** Stk lvls:   1
2980                **
2981                ** History:
2982                **
2983                **    Date     Programmer              Modification
2984                **  --------   ----------   ----------------------------------
2985                **  10/18/83   B.S.         Added documentation
2986                **
2987                ********************************************************************
2988                ********************************************************************
2989 18C42 133  =DEST+  AD1EX
2990 18C45 131         D1=A
2991 18C48 7000        GOSUB  =dest
2992 18C4C 131         D1=A
2993 18C4F 01          RTN
2994                *-
2995                *-
2996 18C51 8000 =atnclr GOVLNG =ATNCLR
           000
2997 18C58             END
```

```
 -LINE   Ext                  -  2610
=ADHEAD  Abs   98743 #1817B7  -   949  1966
 ATNCLR  Ext                  -  2996
=AVE=C   Abs  101307 #18BBB  -  2843
=AVE=D1  Abs  101304 #18BB8  -  2842  2170  2254  2803
 AVMEME  Ext                  -  1893  2309  2362  2417  2843
 AVMEMS  Ext                  -  2250
=AVMMOV  Abs  101322 #18BCA  -  2885  2279
 B2C95   Abs   97942 #17E96  -   309   283   298
 BF2S10  Abs   99954 #18672  -  1957  1963
 BF2S20  Abs   99975 #18687  -  1964  1960
 BF2S30  Abs   99977 #18689  -  1965  1811
=BF2ST+  Abs   99936 #18660  -  1950
=BF2STK  Abs   99939 #18663  -  1951
 BLANKC  Ext                  -   633
 Blanks  Abs       1 #00001  -    17   713   867   875   981
 CHEDIT  Ext                  -  2134
 CHKMEM  Abs   99607 #18517  -  1441  2164  2751
 CK"ON"  Ext                  -   371  2128
 CKIN03  Abs   99683 #18563  -  1695  1736
 CKIN05  Abs   99705 #18579  -  1700  1724
 CKIN10  Abs   99722 #1858A  -  1706  1690  1734
 CKIN20  Abs   99729 #18591  -  1711  1694
 CKIN25  Abs   99757 #185AD  -  1720  1716
 CKIN30  Abs   99782 #185C6  -  1726  1712
 CKINF+  Abs   99640 #18538  -  1683   527
=CKINF-  Abs   99636 #18534  -  1682
=CKINFO  Abs   99650 #18542  -  1685   612   703
=CLCSTR  Abs   98830 #1820E  -  1047
 CMD1ST  Ext                  -  2127  2564
 CMDFND  Ext                  -  2566
 CMDINI  Ext                  -  2561
 CMMA10  Abs   98194 #17F92  -   564   567
 CMMA20  Abs   98204 #17F9C  -   571   565
 CMMA30  Abs   98220 #17FAC  -   577   574
 CMMA50  Abs   98230 #17FB6  -   581   583
 CMMA60  Abs   98247 #17FC7  -   588   576
 COLLAP  Ext                  -   535
 CRLFCK  Abs   97724 #17DBC  -    75   658
 CRLFOF  Ext                  -  2557
 CSLW9J  Ext                  -   310
 CSLWP   Ext                  -   286
 CSLWP9  Abs   97944 #17E98  -   310   360
 CSRW9J  Ext                  -   293   380
 CSRWP   Ext                  -   291
 CURSFL  Ext                  -  2609
 CURSFR  Ext                  -  2553
 D1=@D1  Abs   99928 #18658  -  1894   691  2691
=D1=AVE  Abs   99921 #18651  -  1893  2276  2595
 D1@POS  Abs   98561 #18101  -   728    86   194   264   560   613
=D1STOR  Abs  101289 #18BA9  -  2803  2426  2955
 D=AVMS  Ext                  -  1971
=DEFPRM  Abs  101141 #18B15  -  2624  2373  2450
=DEST+   Abs  101442 #18C42  -  2989  2197  2953
=DISP    Abs   98099 #17F33  -   494
```

```
 DISP05  Abs   98132 #17F54 -   538   531
 DISP10  Abs   98135 #17F57 -   539   553   652   724
 DISP15  Abs   98167 #17F77 -   551   542   585   589
 DISP1j  Abs   98435 #18083 -   651
 DISP20  Abs   98170 #17F7A -   552
 DISP30  Abs   98182 #17F86 -   559   544
 DISP50  Abs   98442 #1808A -   655   548
 DISP80  Abs   98490 #180BA -   702   657
 DISP90  Abs   98529 #180E1 -   715   708
 DISPDC  Ext                -   492
 DISPP   Ext                -   493
 DISPt   Abs       0 #00000 -    11   494  1682
 DONNA   Ext                -  2496
 DOSCRL  Ext                -   423
=DPART2  Abs   97955 #17EA3 -   356  1699  1717
=DPART3  Abs   98040 #17EF8 -   420   354
 DPOS    Ext                -  1696
 DPRT21  Abs   97974 #17EB6 -   363   378
 DPRT23  Abs   98006 #17ED6 -   373   370
 DPRT25  Abs   98029 #17EED -   380   364   372
 DSFORM  Abs   98817 #18201 -   981   885   891   901
 DSLEEP  Ext                -  2737
=DSP$    Abs   99800 #185D8 -  1790
=DSP$00  Abs   99803 #185DB -  1791  2597
 DSP$10  Abs   99862 #18616 -  1806  1801
 DSP$20  Abs   99875 #18623 -  1810  1807
 DSP$30  Abs   99881 #18629 -  1814  1800  1805
 DSP$40  Abs   99888 #18630 -  1817  1825
 DSP$50  Abs   99909 #18645 -  1824  1822
 DSPBFS  Ext                -  1799  1804
 DSPCHA  Ext                -   374
 DSPFMT  Ext                -  1073
 DSPM20  Abs   99100 #1831C -  1213  1211
 DSPMSK  Ext                -  1797  1802  2547  2554
 ENG10   Abs   99216 #18390 -  1255  1256  1260
 ENG20   Abs   99256 #183B8 -  1270  1266
 ENG30   Abs   99268 #183C4 -  1274  1276
 EOLLEN  Ext                -  1720
 ER:COM  Abs  100596 #188F4 -  2346  2325
 ER:NUM  Abs  100605 #188FD -  2350  2331  2333  2339
=EX-STC  Abs  100445 #1885D -  2290
 EXCP10  Abs  100356 #18804 -  2258  2256
=EXCPAR  Abs  100328 #187E8 -  2250  2203
=EXPEXj  Abs   99624 #18528 -  1449   603   702  2111  2186
 EXPP10  Ext                -  2259
 EXPRj   Abs   98779 #181DB -   961  2293
 EXPRj1  Abs  100456 #18868 -  2293  2952
 EXPSKP  Ext                -  2173
 Except  Abs      12 #0000C -    11   369
 Expr    Ext                -   961
 FINDAj  Ext                -  2140
 FINDDO  Ext                -   540
=FINLIN  Abs  100922 #18A3A -  2547  2158  2565  2614  2620  2736  2746
 FIX     Abs   99304 #183E8 -  1287  1217
 FIX10   Abs   99318 #183F6 -  1292  1294
```

```
 FIX20   Abs   99331 #18403 -  1297  1301
 FIX27   Abs   99383 #18437 -  1316  1314
 FIX35   Abs   99407 #1844F -  1325  1304  1311  1318
 FIX40   Abs   99415 #18457 -  1328  1331
 FIX55   Abs   99426 #18462 -  1332  1327
 FIX60   Abs   99450 #1847A -  1341  1338
 FLTDH   Ext                -   608
 FMTNUM  Abs   98830 #1820E -  1048   886   894   898
 FMTPRP  Abs   98785 #181E1 -   964   880   890
 FNDS05  Abs   98947 #18283 -  1087  1084
 FNDS10  Abs   98958 #1828E -  1091  1094
 FSCI    Abs   99300 #183E4 -  1286  1296  1309  1323
 FSCI?   Abs   99390 #1843E -  1319  1299
 FUNCRO  Ext                -   634   636
 FXSC10  Abs   99077 #18305 -  1205  1203
 FXSCEN  Abs   99050 #182EA -  1195  1117
 GDEFPR  Abs  100882 #18A12 -  2450  2124
=GETAVM  Abs   99917 #1864D -  1892   579  1442  2386  2937
 GETPOS  Abs  100847 #189EF -  2434  2202  2388  2428
 GNXTCR  Ext                -  2393
 GPRMPT  Abs  100863 #189FF -  2442  2120  2570
 INBS    Ext                -  2368  2488
 INP010  Abs  100019 #186B3 -  2102  2105
 INP020  Abs  100080 #186F0 -  2124  2101
 INP030  Abs  100084 #186F4 -  2125  2110
 INP040  Abs  100091 #186FB -  2126  2121
 INP045  Abs  100095 #186FF -  2127  2383
 INP050  Abs  100102 #18706 -  2128  2572
 INP070  Abs  100116 #18714 -  2133  2129  2611
 INP080  Abs  100133 #18725 -  2139  2135
 INP150  Abs  100209 #18771 -  2168  2767
 INP200  Abs  100238 #1878E -  2175  2172
 INP205  Abs  100251 #1879B -  2183  2179
 INP210  Abs  100254 #1879E -  2185  2327
 INP300  Abs  100621 #1890D -  2356  2320
 INP310  Abs  100639 #1891F -  2361  2347  2352
 INP320  Abs  100690 #18952 -  2374  2372
 INPA10  Abs  101080 #18AD8 -  2601  2608
 INPATN  Abs  101052 #18ABC -  2594  2142
 INPB10  Abs  100982 #18A76 -  2565  2577  2579  2581
 INPBOT  Abs  100975 #18A6F -  2564  2152  2154
 INPDWN  Abs  101033 #18AA9 -  2584  2148
=INPEND  Abs  101133 #18B0D -  2620  2144
 INPIEX  Abs  101217 #18B61 -  2744  2136
=INP010  Abs  101204 #18B54 -  2738  2617
=INPOFF  Abs  101193 #18B49 -  2736  2156
 INPSTP  Abs  101116 #18AFC -  2614  2130  2604
 INPTDC  Ext                -  2094  2930
 INPTOP  Abs  101043 #18AB3 -  2589  2150
 INPU10  Abs  101021 #18A9D -  2578  2586
 INPU20  Abs  101026 #18AA2 -  2580  2591
 INPUP   Abs  101011 #18A93 -  2575  2146
=INPUT   Abs  100003 #186A3 -  2096  2932
 INPUTP  Ext                -  2095
 InhEOL  Abs       4 #00004 -    22    75   277   421   538   551   651   716
```

```
 InvalE  Abs       0 #00000 -    18  2330
 LINPT+  Abs  101356 #18BEC -  2935  2180
 LINPTP  Ext                -  2931
=LINPUT  Abs  101352 #18BE8 -  2932
 MAKEBF  Ext                -  2159
=MEMERJ  Abs   98176 #17F80 -   556  1446
 MFER42  Ext                -  2738
 MFERR*  Ext                -  2382
 MFWRQ8  Ext                -   595
 MLFFLG  Ext                -  1683  1687
 MOVED3  Ext                -  2891
 MOVEU3  Ext                -  2761
 MAN?    Abs   98799 #181EF -   971   881  1050
 NEGEXP  Abs   99032 #182D8 -  1118  1105
 NINP1O  Abs  100559 #188CF -  2330  2206
 NINP2O  Abs  100586 #188EA -  2340  2336
 NOSCRL  Ext                -  2615
 NXTSTM  Ext                -   688
 NXtstm  Ext                -  2343
 NoCont  Abs      14 #0000E -    11  2616
 NoCur   Abs       3 #00003 -    21
 NumExp  Abs       3 #00003 -    20  2265  2332
 OUT1TK  Ext                -  2273
=PART3   Abs   98455 #18097 -   688
 PCADDR  Ext                -  2690
 POLL    Ext                -  1713  1731
 POP1N   Ext                -   604   776   877
 POP1S   Ext                -  2598
 POPMTH  Ext                -  1450
 PPOS    Ext                -  1722
=PRINT   Abs   98062 #17F0E -   455
 PRINT#  Ext                -   462
=PRINT*  Abs   98103 #17F37 -   527   461
 PRINT.  Abs   98083 #17F23 -   462   459
 PRINTt  Abs       1 #00001 -    11   460
 PRNTDC  Ext                -   453
 PRTP    Ext                -   454
 PUTRE3  Abs   98608 #18130 -   784   789
 PUTRE9  Abs   98627 #18143 -   791   779
=PUTRES  Abs   98581 #18115 -   776   711
=REPROM  Abs  100894 #18A1E -  2488  2126  2571
 RESPTR  Ext                -  2274  2350
 RESREG  Ext                -   778
 REV$    Ext                -   715  2945
 REVPOP  Ext                -  2112
 Return  Abs       0 #00000 -    16   868   874   959  1790  1806  1950  2424
                              2596
 S/N?1O  Abs  100288 #187C0 -  2196  2192  2194
 S/NFlg  Abs       5 #00005 -    23  2189  2195  2204
 SAVSTK  Ext                -   185
 SCI     Abs   99114 #1832A -  1218  1216
 SCI03   Abs   99116 #1832C -  1219  1286
 SCI05   Abs   99158 #18356 -  1234  1229  1232
 SCI10   Abs   99191 #18377 -  1246  1241
 SCI15   Abs   99201 #18381 -  1250  1249
```

```
 SCI20    Abs   99279 #183CF -  1278  1245  1251  1262
 SCI30    Abs   99285 #183D5 -  1281  1283
 SCINj    Abs   99042 #182E2 -  1121  1108
 SEND10   Abs   97779 #17DF3 -   190
=SEND20   Abs   97786 #17DFA -   194   305   645   650
 SEND30   Abs   97796 #17E04 -   197    88
=SENDEL   Abs   97729 #17DC1 -    77   288   588   632
=SENDIT   Abs   97763 #17DE3 -   184   584
=SENDWD   Abs   97813 #17E15 -   259   295   723
 SETTRC   Ext                 -  2185  2951
 SINP10   Abs  100370 #18812 -  2265  2205
 SINP20   Abs  100392 #18828 -  2273  2269  2340
=SINP30   Abs  100460 #1886C -  2296  2289
 SINP40   Abs  100497 #18891 -  2309  2429
 SNDW09   Abs   97868 #17E4C -   283   271
 SNDW11   Abs   97905 #17E71 -   293   307
 SNDW17   Abs   97916 #17E7C -   298   278   280
 SNDW19   Abs   97923 #17E83 -   301   269   274
 SNDW21   Abs   97928 #17E88 -   304   299
=SNDWD+   Abs   97823 #17E1F -   263
 STDF10   Abs   99010 #182C2 -  1110  1102  1120
 STDF20   Abs   99012 #182C4 -  1111  1113
 STDF30   Abs   99029 #182D5 -  1117  1082
 STKC10   Abs   99620 #18524 -  1446  1432
=STKCH+   Abs   99591 #18507 -  1430   957  2943
=STKCHR   Abs   99588 #18504 -  1429   581   893   896   900   984  1054  1065
                             1086  1367  1377  1382  1809  1823  1961  2409
 STKLST   Abs   98853 #18225 -  1054  1058  1069
 STMTDO   Ext                 -  2199  2313
 STMTRO   Ext                 -    77   198   690   728  1700
=STMTST   Abs  101152 #18B20 -  2690  2097  2171  2370  2443
 STORE    Ext                 -  2291  2805
=STR$     Abs   98646 #18156 -   874
 STR$.5   Abs   98684 #1817C -   885   882
=STR$00   Abs   98652 #1815C -   876   869
 STR$01   Abs   98688 #18180 -   886   884
 STR$05   Abs   98870 #18236 -  1062  1052
 STR$10   Abs   98883 #18243 -  1066  1063
 STR$30   Abs   98739 #181B3 -   901   887
 STR$40   Abs   98763 #181CB -   956
 STR$CP   Abs   98696 #18188 -   890   879
 STR$RT   Abs   98636 #1814C -   868   714
=STR$SB   Abs   98633 #18149 -   867
 String   Abs       1 #00001 -    19
 TAB00    Abs   98286 #17FEE -   602   546
 TAB10    Abs   98319 #1800F -   611   599
 TAB20    Abs   98349 #1802D -   621   619   624
 TAB30    Abs   98359 #18037 -   627   622
 TAB40    Abs   98376 #18048 -   633   629
 TAB50    Abs   98394 #1805A -   636   646
 TAB60    Abs   98429 #1807D -   649   642
 TABWRN   Abs   98255 #17FCF -   592   605   607   610
 TOAS05   Abs   99506 #184B2 -  1361  1358
 TOAS10   Abs   99508 #184B4 -  1362  1350  1370
 TOAS15   Abs   99523 #184C3 -  1367  1373
```

```
 TOAS17  Abs   99527 #184C7 -  1368  1352  1386
 TOAS20  Abs   99546 #184DA -  1374  1364
 TOAS30  Abs   99576 #184F8 -  1383  1380
 TOASCI  Abs   99456 #18480 -  1343  1285
 UN"D10  Abs  100747 #1898B -  2397  2412
 UN"D20  Abs  100755 #18993 -  2400  2413
 UN"D30  Abs  100797 #189BD -  2416  2404  2407
 UN"DST  Abs  100714 #1896A -  2386  2262
 USING   Ext                -   532
 Un"dSt  Abs  100366 #1880E -  2262  2266  2272
 XXHEAD  Ext                -  2946
=ZERBUF  Abs  101152 #18B20 -  2649  2125
 adhead  Abs   99980 #1868C -  1966  2425  2949
=atnclr  Abs  101457 #18C51 -  2996  2594
 cmdini  Abs  100968 #18A68 -  2561  2575  2584  2589
=collap  Abs   98125 #17F4D -   535   539  2133
 d=avms  Abs   99987 #18693 -  1971   876  1892
 d=avs+  Abs   99984 #18690 -  1970  1795  1954
 dest    Ext                -  2991
 eIVTAB  Ext                -   594
 eNUMIN  Ext                -  2351
 eTOOFI  Ext                -  2346
 eTOOMI  Ext                -  2360
 expexc  Ext                -  1449
=inexit  Abs  100590 #188EE -  2343  2359  2621  2956
 kcATTN  Ext                -  2141
 kcBOT   Ext                -  2151
 kcCONT  Ext                -  2143
 kcDOWN  Ext                -  2147
 kcLAST  Ext                -  2153
 kcOFF   Ext                -  2155
 kcTOP   Ext                -  2149
 kcUP    Ext                -  2145
 mferrs  Ext                -  2739
 moved3  Abs  101336 #18BD8 -  2891  2166
 nomem   Ext                -   556
 oEOLC1  Abs      12 #0000C -    30
 oEOLLN  Abs      11 #0000B -    28    77
 oHNDLR  Abs       1 #00001 -    26   198   690
 oPOSIT  Abs       6 #00006 -    27   728
 pPRTCL  Abs      14 #0000E -    11  1732
 pPRTIS  Abs      15 #0000F -    11  1714
=popmth  Abs   99630 #1852E -  1450  2201  2296  2427  2954
 store   Abs  101297 #18BB1 -  2805
 tCOMMA  Ext                -   543  2267  2318  2334
 tEOL    Ext                -  2270  2337
 tLINPT  Ext                -  2177
 tSEMIC  Ext                -   541  2108
 tTAB    Ext                -   545
 tUSING  Ext                -   529
```

Input Parameters

  Source file name is SB&IO::MS

  Listing file name is SB/IO:TI:ML::-1

  Object file name is SB%IO:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    M   M  N   N  &     L       CCC   K   K
 2            *    MM MM  N   N  & &   L      C   C  K  K
 3            *    M M M  NN  N  & &   L      C      K K
 4            *    M M M  N N N   &    L      C      KK
 5            *    M   M  N  NN  & & &  L      C      K K
 6            *    M   M  N   N  &  &   L      C   C  K  K
 7            *    M   M  N   N   && &  LLLLL   CCC   K   K
 8            *
 9                 TITLE  LOCK Module <831216.1454>
10 18C58          ABS    #18C58
11            ********************************************************************
12            ********************************************************************
13            **
14            ** Name:    LOCK    -  LOCK Statement Execute
15            **
16            ** Category:   STEXEC
17            **
18            ** Purpose:
19            **      LOCK statement execute.
20            **
21            ** Entry:
22            **      Jumped on LOCK token.
23            **
24            ** Exit:
25            **      NXTSTM.
26            **
27            ** Calls:      EXPEXC, REVPOP
28            **
29            ** Algorithm:
30            **      Evaluates and pops string expression.
31            **      If greater than 8 chars, uses first 8 chars.
32            **      If less than 8 chars, right-pads with nulls.
33            **      Stores result in LOCKWD.
34            **
35            ** History:
36            **
37            **     Date     Programmer            Modification
38            **   --------   ----------   --------------------------------
39            **   05/20/82   NM           Added documentation
40            **
41            ********************************************************************
42            ********************************************************************
43 18C58 0000          REL(5) =DELAYd
         0
44 18C5D 0000          REL(5) =STRNGP
         0
45 18C62 7000 =LOCK    GOSUB  =EXPEXj        Evaluate parameter
46 18C66 8F00          GOSBVL =REVPOP        Reverse chars and POP
         000
47 18C6D AF6           C=A    W              String length
48 18C70 AF0           A=0    W              Room for lockword
49 18C73 CE            C=C-1  A
50 18C75 431           GOC    UNLOCK         Go if argument null
51 18C78 20            P=     0
52 18C7A 80F0          CPEX   0
```

```
     53 18C7E 8AA          ?C=0   A              Expression <= 8 chars?
     54 18C81 40           GOYES  LOCK10         Yes
     55 18C83 2F           P=     15             No.  will take 8
     56 18C85 1531  LOCK10 A=DAT1 WP             Read lockword
     57 18C89 1F00  UNLOCK D1=(5) =LOCKWD
           000
     58 18C90 1517         DAT1=A W              Write lockword
     59 18C94 6000         GOTO   =inexit        GOTO NXTSTM
     60               ****************************************************************
     61               ****************************************************************
     62               **
     63               ** Name:    LOCKD?  -  Password Processing
     64               **
     65               ** Category:   CONFIG
     66               **
     67               ** Purpose:
     68               **      Require user to unlock machine.
     69               **
     70               ** Entry:
     71               **      None.
     72               **
     73               ** Exit:
     74               **      Carry set if machine is unlocked:
     75               **          a) flTNOF was already cleared,
     76               **          b) password was null, or
     77               **          c) user entered correct password.
     78               **        In all above cases, =flTNOF is cleared upon exit.
     79               **      Carry clear if user failed to unlock machine.
     80               **      P=0.
     81               **
     82               ** Calls:      CHEDIT, DSP$00, FINLIN, REVPOP, SFLAG?, SFLAGC,
     83               **             atnclr, bf2dsp, crlfnd, r<rst2, rst2<r.
     84               **
     85               ** Uses.......
     86               **             A,B,C,D,P,D0,D1,R0,R3,ST
     87               **
     88               ** Stk lvls:   4
     89               **
     90               ** Detail:
     91               **      Password processing must be done as part of deep sleep
     92               **      wakeup.  If on-timer and polling failed to clear
     93               **      =flTNOF, password processing is called.
     94               **
     95               ** Algorithm:
     96               **      If turnoff flag clear goto 2.
     97               **      If password=null goto 1.
     98               **      Prompt for password.
     99               **      CHEDIT.
    100               **      If returned with ENDLN and readable chars in display
    101               **        match password goto 1.
    102               **      Return CC.
    103               **    1: Clear =flTNOF.
    104               **    2: RTNSC.
    105               **
    106               ** History:
```

```
107                **
108                **     Date      Programmer            Modification
109                **   --------    ----------   ------------------------------
110                **   05/20/82    NM           Added documentation
111                **
112                ********************************************************************
113                ********************************************************************
114 18C98 20     =LOCKD? P=     0
115 18C9A 04             SETHEX
116 18C9C 3100           LC(2)  =flTNOF
117 18CA0 8E00           GOSUBL =SFLAG?         Turnoff flag set?
          00
118 18CA6 5C1            GONC   TRNON1          No.  return.
119 18CA9 1F00           D1=(5) =LOCKWD
          000
120 18CB0 1537           A=DAT1 W
121 18CB4 97C            ?A#0   W               Is there a lockword?
122 18CB7 E0             GOYES  LCK03           Yes
123 18CB9 3100   TRNON   LC(2)  =flTNOF
124 18CBD 8E00           GOSUBL =SFLAGC         Clear turnoff flag
          00
125 18CC3 02     TRNON1  RTNSC
126 18CC5        LCK03
127 18CC5 8E00           GOSUBL =r<rst2         Save 3 subroutine levels
          00
128 18CCB 8E00           GOSUBL =crlfnd
          00
129 18CD1 1F85           D1=(5) =LCKMSG
          D81
130 18CD8 8E00           GOSUBL =bf2dsp         Put up "password? " message
          00
131 18CDE 8E00           GOSUBL =CHEDIT         Call character editor
          00
132 18CE4 5B0            GONC   FINISH          Return if immediate xeq key
133 18CE7 31D0           LC(2)  13
134 18CEB 962            ?A=C   B               Endln?
135 18CEE 21             GOYES  LCK10           Yes
136 18CF0 7000   FINISH  GOSUB  =atnclr         Clear ATTN flag in case ATTN hit
137 18CF4 7000           GOSUB  =FINLIN
138 18CF8 8E00           GOSUBL =rst2<r         Restore 3 subroutine levels
          00
139 18CFE 03             RTNCC                  No
140 18D00 1F00   LCK10   D1=(5) =AVMEME
          000
141 18D07 147            C=DAT1 A
142 18D0A 135            D1=C                   Point at mathstack
143 18D0D 850            ST=1   0               For DSP$00 to return
144 18D10 7000           GOSUB  =DSP$00         Get password from display
145 18D14 8F00           GOSBVL =REVPOP         Reverse and POP
          000
146 18D1B D6             C=A    A               String length
147 18D1D CE             C=C-1  A
148 18D1F CE             C=C-1  A
149 18D21 CE             C=C-1  A
150 18D23 4CC            GOC    FINISH          Go if input null
```

```
151 18D26 80F0          CPEX   0
152 18D2A 8AA           ?C=0   A              Length <= 8 chars?
153 18D2D 40            GOYES  LCK20          Yes
154 18D2F 2F            P=     15             No.  use 8 chars.
155 18D31 AF0   LCK20   A=0    W
156 18D34 1531          A=DAT1 WP             Read user's input
157 18D38 1B00          DO=(5) =LOCKWD
          000
158 18D3F 1567          C=DAT0 W              Correct lockword
159 18D43 20            P=     0
160 18D45 976           ?A#C   W              Match?
161 18D48 8A            GOYES  FINISH         No
162 18D4A 7000          GOSUB  =FINLIN        Yes.  Clear display.
163 18D4E 8E00          GOSUBL =rst2<r
          00
164 18D54 646F          GOTO   TRNON
165 18D58 B1    =LCKMSG CON(2) #1B            Esc
166 18D5A C3            NIBASC \<\            (cursor off)
167 18D5C 0716          NIBASC \password\     (message)
          3737
          77F6
          2746
168 18D6C F302          NIBASC \? \
169 18D70 FF            NIBHEX FF             End of message
170 18D72               END
```

AVMEME	Ext		-	140				
CHEDIT	Ext		-	131				
DELAYd	Ext		-	43				
DSP$00	Ext		-	144				
EXPEXj	Ext		-	45				
FINISH	Abs	101616 #18CF0	-	136	132	150	161	
FINLIN	Ext		-	137	162			
LCK03	Abs	101573 #18CC5	-	126	122			
LCK10	Abs	101632 #18D00	-	140	135			
LCK20	Abs	101681 #18D31	-	155	153			
=LCKMSG	Abs	101720 #18D58	-	165	129			
=LOCK	Abs	101474 #18C62	-	45				
LOCK10	Abs	101509 #18C85	-	56	54			
=LOCKD?	Abs	101528 #18C98	-	114				
LOCKWD	Ext		-	57	119	157		
REVPOP	Ext		-	46	145			
SFLAG?	Ext		-	117				
SFLAGC	Ext		-	124				
STRNGP	Ext		-	44				
TRNON	Abs	101561 #18CB9	-	123	164			
TRNON1	Abs	101571 #18CC3	-	125	118			
UNLOCK	Abs	101513 #18C89	-	57	50			
atnclr	Ext		-	136				
bf2dsp	Ext		-	130				
crlfnd	Ext		-	128				
flTNOF	Ext		-	116	123			
inexit	Ext		-	59				
r<rst2	Ext		-	127				
rst2<r	Ext		-	138	163			

Input Parameters

  Source file name is MN&LCK::MS

  Listing file name is MN/LCK:TI:ML::-1

  Object file name is MN%LCK:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1              *       SSS   CCC   &     SSS  U   U  BBBB
 2              *      S   S C   C & &   S   S U   U  B   B
 3              *      S     C     & &   S     U   U  B   B
 4              *       SSS  C      &     SSS  U   U  BBBB
 5              *          S C     & & &     S U   U  B   B
 6              *      S   S C   C & & &  S   S U   U  B   B
 7              *       SSS   CCC   && &  SSS   UUU   BBBB
 8              *
 9
10                     TITLE  SUBprogram and DEF FN <831216.1610>
11 18D72               ABS    #18D72
12                     RDSYMB TI%EQU::MS
13                     RDSYMB SB%RAM::MS
```

```
14                EJECT
15              ****************************************************************
16              ****************************************************************
17              **
18              ** Name:(S) pCALSV  -  Poll to save local environment on CALL
19              **
20              ** Category:   POLL
21              **
22              ** Type:       POLL
23              **
24              ** Purpose:
25              **      Give any LEX file a chance to save its local environment
26              **      when CALL is executed.
27              **
28              ** Should poll be "Handled" (return with XM=0)?:
29              **      Since this poll is intended to reach every LEX file, so
30              **      XM should always set to 1 on return.
31              **
32              ** Meaning of "Handling" Poll (what does code do if handled?):
33              **      A LEX file can put a block of its local environment on
34              **      top of the stack. When the ENDSUB is executed later on,
35              **      the LEX file can use this block to restore its local
36              **      environment.
37              **
38              ** Entry conditions for handler (registers, ST, RAM, etc.):
39              **      B[A] = Poll number.
40              **      HEX mode.
41              **      P=0.
42              **      AVMEME(available memory end) is pointing at current top
43              **      of stack.
44              **
45              ** Normal exit conditions from handler if handled (ST, RAM,
46              ** registers, etc.):
47              **      Carry clear (POLL only).
48              **      HEX mode.
49              **      XM=1.
50              **      Update the AVMEME to point at the top of the block which
51              **      just been put on to the top of the satck.
52              **
53              ** Normal exit conditions from handler if not handled (ST, RAM,
54              ** registers, etc.):
55              **      Carry clear (POLL only).
56              **      HEX mode.
57              **      XM=1.
58              **
59              ** Error exit conditions from handler (POLL only):
60              **      Carry set.
61              **      HEX mode.
62              **      C[3,0] = Error code
63              **
64              ** Available subroutine levels:
65              **      5
66              **
67              ** NOTE:
68              **    Definition of the save block:  (starting from lower addr.)
```

```
69          **
70          **      Nibs    Meaning
71          **      ------  ----------------------------------------
72          **      1-2     LEX file ID
73          **      3-5     Block length in nibs(not include the 1st 5 nibs)
74          **      6       # of update addresses following
75          **      7-n     Update addresses 5 nibs each
76          **      n-m     Anything else
77          **
78          ** What registers/RAM may be used if handled?:
79          **      A-D, D0, D1, P ,R0-R3, ST, scratch RAM
80          **
81          ** What registers/RAM may be used if not handled?:
82          **      A-D, D0, D1, P ,R0-R3, ST, scratch RAM
83          **
84          ** What registers/RAM may be used if error exit (POLL only)?:
85          **      A-D, D0, D1, P, R0-R3, ST, scratch RAM
86          **
87          ** Special memory/pointer considerations (are pointers funny?):
88          **      None.
89          **
90          ** Envisioned application(s):
91          **      Allows a LEX file to stack and unstack local data that
92          **      is not stored in a system buffer.  This may be useful
93          **      to applications which can be called recursively, since
94          **      system buffers are global and are not allocated recur-
95          **      sively.
96          **
97          ** History:
98          **
99          **    Date     Programmer            Modification
100         **  --------   ----------   ------------------------------------
101         **  04/18/83   SC           Document
102         **
103         ********************************************************************
104         ********************************************************************
105         *
106         ********************************************************************
107         ********************************************************************
108         **
109         ** Name:(S) pCALRS  -  Poll  to restore local environment
110         **
111         ** Category:   POLL
112         **
113         ** Type:       POLL
114         **
115         ** Purpose:
116         **      Give any LEX file a chance to restore its local
117         **      environment when ENDSUB is executed.
118         **
119         ** Should poll be "Handled" (return with XM=0)?:
120         **      Since this poll is intended to reach every LEX file, so
121         **      XM should always set to 1 on return.
122         **
123         ** Meaning of "Handling" Poll (what does code do if handled?):
```

```
124             **      A LEX file can restore its local environment saved at
125             **      CALL time (by respond to pCALSV poll)
126             **
127             ** Entry conditions for handler (registers, ST, RAM, etc.):
128             **      B[A] = Poll number.
129             **      HEX mode.
130             **      P=0.
131             **      CALSTK is pointing at the first of all the save blpcks.
132             **
133             ** Normal exit conditions from handler if handled (ST, RAM,
134             ** registers, etc.):
135             **      Carry clear (POLL only).
136             **      HEX mode.
137             **      XM=1.
138             **
139             ** Normal exit conditions from handler if not handled (ST, RAM,
140             ** registers, etc.):
141             **      Carry clear (POLL only).
142             **      HEX mode.
143             **      XM=1.
144             **
145             ** Error exit conditions from handler (POLL only):
146             **      Carry set.
147             **      HEX mode.
148             **      C[3,0] = Error code
149             **
150             ** Available subroutine levels:
151             **      3
152             **
153             ** NOTE:
154             **      How to find the save block of your own :
155             **
156             **      Starting from the CALSTK, look for first 2 nibbles of
157             **      each block for your LEX ID. All the save blocks are
158             **      link listed. When your block is found, just use the
159             **      information to restore your local environment, don't
160             **      collapse the block. All the update addresses in the
161             **      block are justified if memory had been moved.
162             **
163             ** What registers/RAM may be used if handled?:
164             **      A-D, D0, D1, P ,R0-R3, ST, scratch RAM
165             **
166             ** What registers/RAM may be used if not handled?:
167             **      A-D, D0, D1, P ,R0-R3, ST, scratch RAM
168             **
169             ** What registers/RAM may be used if error exit (POLL only)?:
170             **      A-D, D0, D1, P, R0-R3, ST, scratch RAM
171             **
172             **
173             ** History:
174             **
175             **    Date     Programmer             Modification
176             **  --------   ----------   ------------------------------------
177             **  04/18/83   SC           Document
178             **
```

```
179          ******************************************************************
180          ******************************************************************
181          *
182          ******************************************************************
183          ******************************************************************
184          **
185          ** Name:(S) pFNIN   -  Poll at start of multiline U.D.F.
186          **
187          ** Category:   POLL
188          **
189          ** Type:       FPOLL
190          **
191          **
192          ** Purpose:
193          **      Poll before start execution of a multiline user-defined
194          **      function.
195          **
196          ** Should poll be "Handled" (return with XM=0)?:
197          **      If handled set XM=1 on return.
198          **
199          ** Meaning of "Handling" Poll (what does code do if handled?):
200          **      This poll give everybody a chance to do something, so
201          **      the poller doesn't care it will be handled or not.
202          **
203          ** Entry conditions for handler (registers, ST, RAM, etc.):
204          **      Carry set on entry.
205          **      B[A] = Poll number.
206          **      HEX mode.
207          **      P=0.
208          **
209          ** Normal exit conditions from handler if handled (ST, RAM,
210          ** registers, etc.):
211          **      HEX mode.
212          **      XM=1.
213          **
214          ** Normal exit conditions from handler if not handled (ST, RAM,
215          ** registers, etc.):
216          **      HEX mode.
217          **      XM=1.
218          **
219          ** Available subroutine levels: 4
220          **
221          ** What registers/RAM may be used if handled?:
222          **      Everything but the R1
223          **
224          ** What registers/RAM may be used if not handled?:
225          **      A-C, D[15-5] D0, D1, P
226          **      --NOTE: D[A] is sacred
227          **      R1-R4, ST, scratch RAM
228          **
229          **
230          ** History:
231          **
232          **    Date     Programmer              Modification
233          **  --------   ----------   ------------------------------------
```

```
234             ** 05/10/83   SC          Document
235             **
236             ************************************************************************
237             ************************************************************************
238             *
239             ************************************************************************
240             ************************************************************************
241             **
242             ** Name:(S) pFNOUT  -  Poll at end of multiline U.D.F.
243             **
244             ** Category:   POLL
245             **
246             ** Type:       FPOLL
247             **
248             **
249             ** Purpose:
250             **      Poll before exiting  a multiline user-defined
251             **      function.
252             **
253             ** Should poll be "Handled" (return with XM=0)?:
254             **      If handled set XM=1 on return.
255             **
256             ** Meaning of "Handling" Poll (what does code do if handled?):
257             **      This poll give everybody a chance to do something, so
258             **      the poller doesn't care it will be handled or not.
259             **
260             ** Entry conditions for handler (registers, ST, RAM, etc.):
261             **      Carry set on entry.
262             **      B[A] = Poll number.
263             **      HEX mode.
264             **      P=0.
265             **
266             ** Normal exit conditions from handler if handled (ST, RAM,
267             ** registers, etc.):
268             **      HEX mode.
269             **      XM=1.
270             **
271             ** Normal exit conditions from handler if not handled (ST, RAM,
272             ** registers, etc.):
273             **      HEX mode.
274             **      XM=1.
275             **
276             ** Available subroutine levels: 4
277             **
278             ** What registers/RAM may be used if handled?:
279             **      Everything but the R0
280             **
281             ** What registers/RAM may be used if not handled?:
282             **      A-C, D[15-5] D0, D1, P
283             **      --NOTE: D[A] is sacred
284             **      R1-R4, ST, scratch RAM
285             **
286             **
287             ** History:
288             **
```

```
289             **    Date      Programmer              Modification
290             **  --------    ----------   ------------------------------------
291             **  05/10/83    SC           Document
292             **
293             ********************************************************************
294             ********************************************************************
```

```
295                     EJECT
296                *
297                * SUB - EXECUTION OF SUB STATEMENT STOP THE PROGRAM RUNNING.
298                *
299 18D72 0000          REL(5) =SUBDC
          0
300 18D77 0000          REL(5) =SUBP
          0
301 18D7C 8D00 =SUB     GOVLNG =END
          000
302                *
303 18D83 1FB4  D1PRCT  D1=(5) =PRMCNT
          9F2
304 18D8A 01            RTN
305                *
306                ********************************************************************
307                ********************************************************************
308                **
309                ** Name:(S) CALL    -  Sub-program call execution
310                **
311                ** Category:   STEXEC
312                **
313                ** Purpose:  Call a sub-program
314                **
315                ** Enry: DO pts past the tCALL token
316                **
317                ** Exit:
318                **      To NXTSTM if successful
319                **      To BSERR  if error
320                **
321                ** Calls:  I/OALL, GETC#-, FDCH#, EXPEXC, DEST, NEWVAR, SCHSUB
322                **         LNSKP-, TRFLCK, TRCLIN, TRTOEN, I/OFND, EXPCH#, FNDMK-
323                **         POPCH#, CR-VAR, POLL, STRASN, MOVEMD, FSPECx, FINDF
324                **         SFLAGC, SFLAGS, SFLAG?, GETSTC, PRSCOO, CHKSPC
325                **
326                ** Uses:  Everything
327                **
328                ** Stk lvls:  All
329                **
330                ** Detail:
331                **
332                ** 1. Search the subprogram and save the name on stack.
333                ** 2. Start process the actuall parameters:
334                **  a. Go down the parameters list, call expression to get
335                **     every parameters.
336                **  b. Save the value or the address of each parameter on the
337                **     stack. Put a cap on top of each parameter to indicate
338                **     it is a vaule or an address. (Parse routine already
339                **     figured out each parameter is passed by value or by
340                **     reference).
341                **  c. If find an "#" sign preceding an expression, it must be
342                **     a channel number. Then make sure the channel is open,
343                **     also put a cap to indicate this is a channel number.
344                **  d. Call the routine DEST right after returning from the
345                **     expression execution routine. If the parameter is a non-
```

```
346           **     existent variable, call the routine NEWVAR to create the
347           **     variable. THEN collapse the stack(except the subprogram
348           **     name), process the actual parameters all over again
349           **     starting from the beginning. The reason for starting from
350           **     the beginning is that some of the references that already
351           **     been processed may need to be adjusted due to the
352           **     creation of new variable. In order to save code, I choose
353           **     to re-evaluate the all the expression rather than only to
354           **     adjust those references.
355           **
356           ** 3. Save the calling environment on the stack(on top of the
357           **    actual parameters information).
358           **     (lowest address):
359           **
360           **     0004F  (5 nibs):  ID & length
361           **     A      (1)     :  Update pointers count
362           **                    :  Following 10 pointers are absolute
363           **     CURRST (5)         addresses, they will be adjusted
364           **     PRGMST (5)         when memory moved.
365           **     PRGMEN (5)
366           **     CURREN (5)
367           **     PCADDR (5)
368           **     CNTADR (5)
369           **     ERRSUB (5)
370           **     ERRADR (5)
371           **     ONINTR (5)
372           **     DATPTR (5)
373           **
374           **     Offset to previous FORSTK (5)
375           **              //       GSBSTK (5)
376           **              //       ACTIVE (5)
377           **              //       CALSTK (5)
378           **
379           **
380           ** 4. A LEX file can save its local environment on the call
381           **    stack too. At this point, a poll(pCALSV) will be issued.
382           **    An LEX file when answering to this poll can put a save
383           **    block on top of the current stack pointer(pointed by D1).
384           **    The format of the save block is as follow :
385           **
386           **     nibs    meaning
387           **     ----    ----------------------------------------
388           **     1-2     LEX file ID.
389           **     3-5     Save block length(exclude the first 5 nibs).
390           **     6       Number of addresses follow that need to be
391           **             adjusted when memory moved.
392           **     7-11    First address if any.
393           **     .....
394           **     .....   to end of the block.
395           **
396           **  5. Search for the subprogram.
397           **
398           **  6. Set CALSTK, ACTIVE, GSBSTK, FORSTK to the current stack
399           **     pointer(MTHSTK)
400           **
```

```
401             ** 7. Clear the variable chain head table
402             **
403             ** 8. Put a level mark in the channel number assign buffer.
404             **
405             ** 9. Process the formal parameters:
406             **    a. If the parameter is a channel, open the channel.
407             **    b. Call expression execution to get each variable and
408             **       call the routine DEST right after that. Then call
409             **       the routine CR-VAR to create the dope vector of each
410             **       variable.
411             **    c. Dig out the actual parameter from the stack one at a
412             **       time and compare its type with the corresponding
413             **       formal parameter.
414             **    d. Assign value or indirect address to the formal
415             **       parameter.
416             **
417             ** 10. Pull all the actual parameter information from stack and
418             **     adjust all the offset values in the call save block.
419             **
420             ** 11. Clear ERRSUB, ERRADR, ONINTR, DATPTR
421             **
422             ** 12. Execute the subprogram.
423             **
424             ********************************************************************
425             ********************************************************************
426             *
427              Binary EQU   10
428             *
429             ********************************************************************
430             ********************************************************************
431             **
432             ** Name:(S) CALBIN  -  Binary program call BASIC subprogram
433             **
434             ** Category:   STEXEC
435             **
436             ** Purpose:  To allow a binary program to call a BASIC
437             **           subprogram.
438             **
439             ** Entry:  This GOSBVL has to precede right before the CALL
440             **         statement. The binary file has to construct the CALL
441             **         statement exactly as it is in a BASIC file. The first
442             **         two nibs are the statement length and the last two
443             **         nibs are the EOL.
444             **
445             ** Exit:  The execution of the binary program will be resumed
446             **        after CALL statement.
447             **
448             ** Uses:  Everything
449             **
450             ** Stk lvls:  Only one RSTK will be saved, the one calls CALBIN.
451             **
452             ** Note:  When CALBIN is called, the PCADDR will be set to @ the
453             **        line length of the CALL statement.
454             **        When ENDSUB is executed, if it is returning to binary
455             **        code, the PCADDR will be set to @ the end of the
```

```
456                **      CALL statement.
457                **
458                ************************************************************
459                ************************************************************
460                *
461 18D8C 07       =CALBIN C=RSTK
462 18D8E 8EDE             GOSUBL DOPCAD          Set PCADD to DO
           90
463 18D94 144              DAT0=C A
464                *
465 18D97 134              D0=C
466 18D9A 163              D0=D0+ 4
467 18D9D 85A              ST=1   Binary
468 18DA0 6010             GOTO   CAL010
469                *
470 18DA4 0000             REL(5) =CALLDC
           0
471 18DA9 0000             REL(5) =CALLP
           0
472 18DAE 84A      =CALL   ST=0   Binary
473 18DB1 14A       CAL010 A=DAT0 B
474 18DB4 3100             LC(2)  =tEOL
475 18DB8 9E2              ?A<C   B
476 18DBB 80               GOYES  CAL020
477                *
478                * THIS IS CALL WITHOUT SUBPROGRAM NAME
479                *
480 18DBD AF0              A=0    W
481 18DC0 561              GONC   CAL045
482                *
483 18DC3 8F00      CAL020 GOSBVL =FILXQ^        GET SUBPROGRAM NAME
           000
484 18DCA 460              GOC    CAL040        NO ERROR IF CARRY SET
485 18DCD 6853      CAL030 GOTO   MISUBP        SAY "SUBPROGRAM NOT FOUND"
486                *
487 18DD1 B47       CAL040 D=D+1  S
488 18DD4 58F              GONC   CAL030        IF D(S) IS NOT F, ERROR
489 18DD7 AF8       CAL045 B=A    W             B = SUBPROGRAM NAME
490 18DDA 8F00             GOSBVL =GETST        SAVE STATUS BITS
           000
491 18DE1 7877             GOSUB  D1FSTK        SET MTHSTK = FORSTK - 16
492 18DE5 1CF              D1=D1- 16
493 18DE8 8E00             GOSUBL =CHKSTK       STILL ROOM ON STACK ?
           00
494 18DEE 460              GOC    CAL050
495 18DF1 6474             GOTO   CALNOR
496                *
497 18DF5 8E00      CAL050 GOSUBL =SAVEDO       SAVE PC IN STMTDO
           00
498 18DFB AF4              A=B    W
499 18DFE 1517             DAT1=A W             SAVE SUBPROGRAM NAME ON STACK
500                *
501 18E02 316D             LC(2)  =flNOFN       Disallow UDF in parameters
502 18E06 8E00             GOSUBL =SFLAGS
           00
```

```
503                *
504 18E0C 737F  CAL060 GOSUB  D1PRCT        SET PARAMETER COUNT TO ZERO
505 18E10 D0           A=0    A
506 18E12 149          DAT1=A B
507 18E15 7447         GOSUB  D1FSTK        SET MTHSTK = FORSTK-16
508 18E19 1CF          D1=D1- 16
509 18E1C 137          CD1EX
510 18E1F 1C4          D1=D1- 5
511 18E22 145          DAT1=C A
512 18E25 137          CD1EX
513                *
514 18E28 14A          A=DAT0 B
515 18E2B 8F00         GOSBVL =EOLXCK       SEE IF CALL WITHOUT NAME
           000
516 18E32 402          GOC    CAL105        IF SO, GOTO CAL105
517 18E35 161          D0=D0+ 2
518 18E38 3100         LC(2)  =tPRMST       IF NEXT TOKEN IS NOT AN "tPRMST"
519 18E3C 966          ?A#C   B              MEANS NO PARAMETER PASSING
520 18E3F 41           GOYES  CAL105
521 18E41 551          GONC   CAL110        (B.E.T.)
522 18E44 14A   CAL100 A=DAT0 B             READ NEXT TOKEN
523 18E47 3100         LC(2)  =tPRMEN
524 18E4B 966          ?A#C   B              ARE WE AT END OF LINE?
525 18E4E 90           GOYES  CAL110        CONTINUE IF WE ARE NOT
526 18E50 161          D0=D0+ 2
527 18E53 6E61  CAL105 GOTO   CAL400
528 18E57 136   CAL110 CD0EX                SAVE PC IN RSTK
529 18E5A 06           RSTK=C
530 18E5C 136          CD0EX
531                *
532                * Set the NO UDF flag before calling expression execution
533                *
534                *
535                * CHECK IF THE NEXT PARAMETER IS A CHANNEL #
536                *
537 18E5F 14A          A=DAT0 B
538 18E62 3132         LCASC  \#\
539 18E66 966          ?A#C   B
540 18E69 C3           GOYES  CAL112         IF NOT A CHANNEL # GOTO CAL112
541 18E6B 161          D0=D0+ 2
542 18E6E 8E00         GOSUBL =GETC#-        GET THE CHANNEL #
           00
543 18E74 76D6         GOSUB  D1MSTK         D1 PTS TO MTHSTK
544 18E78 D6           C=A    A              SHIFT CHANNEL # LEFT 1 NIBBLE
545 18E7A BF2          CSL    W
546 18E7D 1557         DAT1=C W
547 18E81 AE8          B=A    B
548 18E84 136          CD0EX
549 18E87 108          R0=C                  SAVE PC IN R0
550 18E8A 8E00         GOSUBL =FDCH#         MAKE SURE THE CHANNEL # EXIST
           00
551 18E90 5E0          GONC   CAL111         CHANNEL # NOT FOUND
552 18E93 118          C=R0
553 18E96 134          D0=C
554 18E99 850          ST=1   0              SET FLAG TO INDICATE CHANNEL #
```

```
555 18E9C 411          GOC    CAL114        (B.E.T.)
556 18E9F 8C00  CAL111 GOLONG =F#NFND
          00
557             *
558             *
559 18EA5 8E5C  CAL112 GOSUBL expr          CALL EXPRESSION EXECUTION ROUTINE
          11
560             *
561             *
562 18EAB 840          ST=0   0
```

```
563                     EJECT
564                *
565                * IF NEXT TOKEN IS "E0" (PASS BY REFERENCE), WE HAVE THE
566                * FOLLOWING INFORMATION:
567                *-------------------------------------------------------------
568                *I VARIABLE FOUND(B(A)>h1000)   I VARIABLE NOT FOUND B(A)<=h1000
569                *I-----------------------------I-----------------------------
570                *I B(S)= 0,1,2: REAL,SHORT,INT. I B(S)= 0:  REAL ONLY
571                *I B(A)= VARIALBE ADDRESS       I B(X)= VARIABLE NAME
572                *I STACK= THE NUMBER IN REAL    I STACK= ONE REG. OF ZERO
573                *I-----------------------------I-----------------------------
574                *I B(S)= 8:  NUMERIC ARRAY      I B(S)=8:  NUMERIC ARRAY
575                *I B(14)= SUBSCRIPT COUNT       I (F-R1-3)= SUBSCRIPT COUNT
576                *I B(A)= VAR. DOPE VECTOR ADDR. I B(X)= VARIALBE NAME
577                *I STACK= VAR. DOPE VECTOR      I STACK= C9n000000..(n-Sub.Cnt)
578                *I                             I IF (F-R1-0)= REG.# ( A(1) )
579                *I                             I    (F-R1-0)= FFFFF ( A() )
580                *I-----------------------------I-----------------------------
581                *I B(S)= D:  STRING VARIABLE    I B(S)= D:  STRING VARIABLE
582                *I B(A)= STRING ADDRESS         I B(A)= VARIABLE NAME
583                *I STACK= F0,LENGTH(5),ADDR(5), I STACK= F0 00000 NAME 0000
584                *I       MAX.LEN(4),(STRING...) I       (2)  (5)   (5)  (4)
585                *I-----------------------------I-----------------------------
586                *I B(S)= C:  STRING VECTOR      I B(S)= C:  STRING VECTOR
587                *I B(A)= DOPE VECTOR ADDRESS    I B(A)= VARIABLE NAME
588                *I B(14)= SUBSCRIPT COUNT       I (F-R1-3)= SUBSCRIPT COUNT
589                *I STACK= VAR. DOPE VECTOR      I STACK= F9 00000 NAME 0000
590                *I                             I IF (F-R1-0)= REG.# ( A$(1) )
591                *I                             I    (F-R1-0)= FFFFF ( A$() )
592                *I-----------------------------I-----------------------------
593                *I B(S)= E,F:  COMPLEX, S.COMPL I
594                *I B(A)= COMPLEX NUMBER ADDR    I
595                *I STACK= E0 IMAGINARY REAL     I
596                *I (F-R0-0)= DOPE VECTOR ADDR   I
597                *I-------------------------------------------------------------
598                *
599                * IF THE NEXT TOKEN IS "E1" (PASS BY VALUE), WE HAVE THE
600                * FOLLOWING INFORMATION:
601                *-------------------------------------------------------------
602                *I B(S) = 0,1,2:  REAL, SHORT, INTEGER                        I
603                *I STACK= THE NUMBER IN REAL                                  I
604                *I-----------------------------------------------------------I
605                *I B(S) = E, F  :COMPLEX, SHORT COMPLEX                       I
606                *I STACK= E0, IMAGINARY, REAL                                 I
607                *I-----------------------------------------------------------I
608                *I B(S) = D:  STRING                                          I
609                *I STACK= F0, STRING LENGTH IN NIBBLES(5), STRING ADDR(5), MAX.I
610                *I        STRING LENGTH IN BYTES(4), (STRING .....)          I
611                *I-----------------------------------------------------------I
612                *
613                *
614 18EAE 71DE  CAL114 GOSUB  D1PRCT        INCREMENT THE PARAMETER COUNT
615 18EB2 14B          A=DAT1 B             A= CURRENT PARAMETER COUNT
616 18EB5 14E          C=DAT0 B             THE TOKEN FOLLOWS THE PARAM TELLS
617 18EB8 161          D0=D0+ 2             PASS BY REFER.(E0) OR BY VALUE(E1)
```

```
618 18EBB 870          ?ST=1  0              Channel # ?
619 18EBE 70           GOYES  CAL115
620 18EC0 90A          ?C=0   P              IS THIS PARAMETER PASS BY REFER. ?
621 18EC3 60           GOYES  CAL116         YES, IF THE TOKEN IS "E0"
622 18EC5 63A0 CAL115  GOTO   CAL300
623 18EC9 D2   CAL116  C=0    A              SEE IF THE VARIABLE FOUND OR NOT
624 18ECB B56          C=C+1  M
625 18ECE 8B5          ?B<C   A              IF B(A) >= h01000, VARIABLE EXIST
626 18ED1 37           GOYES  CAL200
627                    *
628                    * PASS BY REFERENCE, AND VARIABLE EXIST
629                    *
630 18ED3 B64          A=A+1  B               INCREMENT PARAMETER COUNT
631 18ED6 149          DAT1=A B
632 18ED9 07           C=RSTK                POP THE SAVED DO OUT
633 18EDB 7F66         GOSUB  D1MSTK         RESTORE STACK POINTER
634 18EDF 2E           P=     14
635 18EE1 3102         LCHEX  20
636 18EE5 9C1          ?B>C   S              IF B(S)<=2 THEN THE NUMBER IS REAL
637 18EE8 80           GOYES  CAL120          SHORT OR INTEGER
638                    * SIMPLE VARIABLE:  REAL, SHORT, INTEGER
639 18EEA A81          B=0    P              SET SUBSCRIPT COUNT TO 0
640 18EED 543          GONC   CAL150         (B.E.T.)
641 18EF0 310D CAL120  LCHEX  D0             IF B(S)= 8 OR C, IT IS NUMERIC
642 18EF4 9C5          ?B<C   S               ARRAY OR STRING VECTOR
643 18EF7 B2           GOYES  CAL150
644                    * STRING VARIABLE OR COMPLEX NUMBER
645 18EF9 A81          B=0    P
646 18EFC 171          D1=D1+ 2             PASS THE HEADER OF COMPLEX OR STRING
647 18EFF 941          ?B=C   S              IF B(S)= D, IT IS A STRING VARIABLE
648 18F02 80           GOYES  CAL130
649 18F04 17F          D1=D1+ 16            2+16+16 NIBBLES FOR A COMPLEX NUMBER
650 18F07 5A1          GONC   CAL150        ON STACK, ONLY 16 NEEDED FOR THE REF
651                    * STRING VARIABLE
652 18F0A 174  CAL130  D1=D1+ 5              POINT TO STRING ADDR & MAX.LEN
653 18F0D 15F8         C=DAT1 9
654 18F11 28           P=     8
655 18F13 A9D          BCEX   WP             SAVE ADDR & MAX.LEN IN B(0-8)
656 18F16 1C4          D1=D1- 5
657 18F19 8E00         GOSUBL =POPSTR        PUSH OUT THE STRING
          00
658 18F1F 1CF          D1=D1- 16
659 18F22 AF9  CAL150  C=B    W              WRITE THE REFERENCE ON TO THE STACK
660 18F25 1557         DAT1=C W
661 18F29 1C0          D1=D1- 1              WRITE AN "0" ON TOP OF THE STACK
662 18F2C 20   CAL160  P=     0               TO REMEMBER THIS PARAM PASS BY
663 18F2E A82          C=0    P               REFERENCE
664 18F31 15D0 CAL170  DAT1=C 1
665 18F35 7306         GOSUB  MSTKD1         PUT STACK POINTER TO C
666 18F39 6A0F         GOTO   CAL100         GOTO LOOK AT NEXT PARAMETER
667 18F3D 8D00 =dest   GOVLNG =DEST
          000
668                    * PASS BY REFERENCE BUT VARIABLE NOT EXIST, SO WE HAVE TO CREATE
669                    * THE VARIABLE AND START THE WHOLE PROCESS AGAIN
670 18F44 7606  CAL200 GOSUB  D1MSTK
```

```
671 18F48 71FF          GOSUB  dest          SAVE POINTERS FOR THE CREATE ROUTINE
672 18F4C AF9           C=B    W              GET TYPE AND NAME TO C(S) & C(A)
673 18F4F 8F00          GOSBVL =NEWVAR        CREATE THE NEW VARIABLE
          000
674 18F56 07            C=RSTK                POP OFF THE SAVED PC
675 18F58 560           GONC   CAL210
676 18F5B 6A03          GOTO   CALNOR         ALLOCATION CRASH IF CARRY SET
677 18F5F 8E00  CAL210  GOSUBL =RESTDO        RESTORE PC TO SATRT OF PARAMETERS
          00
678 18F65 66AE          GOTO   CAL060
679             *
680             ****************
681             *
682             * PASS BY VALUE:
683             *
684 18F69 B64   CAL300  A=A+1  B               INCREMENT PARAMETERS COUNT
685 18F6C 149           DAT1=A B
686 18F6F 07            C=RSTK
687 18F71 30A           LCHEX  A
688 18F74 DA            A=C    A              A(0) = "A"
689 18F76 74D5          GOSUB  D1MSTK         ONLY REAL, COMPLEX, OR STRING CAN
690 18F7A 14F           C=DAT1 B                          F - STRING
691 18F7D 986           ?C<A   P                          E - COMPLEX
692 18F80 03            GOYES  CAL330         If is real, O.K.
693             *
694             * If is not real, see if is a array vector
695             *
696 18F82 816           CSRC
697 18F85 90A           ?C=0   P
698 18F88 80            GOYES  CAL310         If Not, check if is string value ?
699             *
700             * Passing an array vector as a value(by enclosing the array name
701             *  BY a pair of parenthesis)
702             *
703 18F8A 8C00          GOLONG =RDATTY
          00
704             *
705 18F90 B46   CAL310  C=C+1  S               Is this a string value ?
706 18F93 5C1           GONC   CAL330         if not, O.K.
707 18F96 171           D1=D1+ 2             STACK = FO STR.LEN(5) ADDR(5) MAX.(4)
708 18F99 143           A=DAT1 A              SEE HOW LONG THE STRING IS
709 18F9C D2            C=0    A              NOT ALLOW PASS A STRING > 18 CHARS
710 18F9E 3104          LCHEX  40
711 18FA2 8BA           ?A<=C  A
712 18FA5 80            GOYES  CAL320
713 18FA7 8C00          GOLONG =STROVF        STRING OVER FLOW
          00
714 18FAD 1C1  CAL320   D1=D1- 2
715 18FB0 1C0  CAL330   D1=D1- 1
716 18FB3 301           LC(1)  1              WRITE AN "1" ON TOP OF STACLK
717 18FB6 860           ?ST=0  0               IS THIS A CHANNEL # ?
718 18FB9 50            GOYES  CAL335
719 18FBB B06           C=C+1  P
720 18FBE 627F  CAL335  GOTO   CAL170
721             *
```

```
                *  END OF ACTUAL PARAMETERS LIST, START PROCESS THE FORMAL
                *  PARAMETER.
                *
                *  THE FOLLOWING POINTERS ARE STORED IN CALSTK
                *
                *       0004F          (5)
                *       A              (1) - # of update addresses
                *       CURRST         (5)
                *       PRGMST         (5)
                *       PRGMEN         (5)
                *       CURREN         (5)
                *       PCADDR         (5)
                *       CNTADR         (5)
                *       ERRSUB         (5)
                *       ERRADR         (5)
                *       ONINTR         (5)
                *       DATPTR         (5)
                *       OFFSET TO PREVIOUS FORSTK    (5)
                *       OFFSET TO PREVIOUS GSBSTK    (5)
                *       OFFSET TO PREVIOUS ACTIVE    (5)
                *       OFFSET TO PREVIOUS CALSTK    (5)
                *       PREVIOUS PARAMETER COUNT     (2)
                *       OFFSET TO PREVIOUS PRMPTR+2  (5)
                *       RETURN TYPE    (1)
                **
                *        TOTAL 84 NIBBLES
                *
                *
18FC2 136     CAL400 CDOEX
18FC5 108            RO=C                    RO = PC
                *
                *  Set PRGMST & PRGMEN only if not running
                *
18FC8 8F00           GOSBVL =RUSUS?          Running or suspended ?
      000
18FCF 4E1            GOC    CAL404           If so, don't set program scope
                *
18FD2 8F00           GOSBVL =GETSTC          See if File type is BASIC
      000
18FD9 5D0            GONC   CAL402           If so, set program scope
18FDC 3340           LC(4)  =fBIN            See if file type is BINARY
      2E
18FE2 8A6            ?A#C   A
18FE5 90             GOYES  CAL404           If not, don't set prog scope
18FE7 8F00    CAL402 GOSBVL =PRSCOO
      000
                *
18FEE D2      CAL404 C=0    A
18FF0 8F00           GOSBVL =CHKSPC          SEE IF STILL ROOM ON STACK
      000
18FF7 560            GONC   CAL410
18FFA 6B62    CAL405 GOTO   CALNOR
18FFE 8E00    CAL410 GOSUBL =rstst           Restore status bits.
      00
19004 1B99           D0=(5) =MTHSTK          SAVE MTHSTK&FORSTK TO S-R0-0&S-R0-1
```

```
          5F2
771 1900B 1F17          D1=(5) =S-R0-0
          8F2
772 19012 7365          GOSUB  SAVE10
773 19016 135           D1=C
774               *
775               * SAVE RETURN TYPE
776               *
777 19019 1C0           D1=D1- 1
778 1901C D2            C=0    A
779 1901E 86A           ?ST=0  Binary
780 19021 50            GOYES  CAL412
781 19023 308           LC(1)  8               8 FOR RETURN TO BINARY FUNCTION
782 19026 87D   CAL412  ?ST=1  13
783 19029 40            GOYES  CAL414
784 1902B E6            C=C+1  A
785               *
786 1902D 1550  CAL414  DAT1=C P
787 19031 8E5F          GOSUBL SAVPRM          SAVE PARAMETER COUNT & POINTER
          01
788 19037 D2            C=0    A
789 19039 14C           DAT0=C B               ZERO CURRENT COUNT
790 1903C 1BDA          D0=(5) =CALSTK         SAVE THE OFFSET OF FORSTK,GSBSTK,
          5F2
791 19043 2C            P=     12               ACTIVE AND CALSTK
792 19045 8EAF  CAL415  GOSUBL SAVPTR
          01
793 1904B 184           D0=D0- 5
794 1904E 0C            P=P+1
795 19050 54F           GONC   CAL415
796 19053 1BD8          D0=(5) =ONINTR         SAVE DATPTR - ERRSUB
          6F2
797 1905A 7A05          GOSUB  SAVE20
798               *
799 1905E 7115          GOSUB  SAV10+          SAVE PCADDR & CNTADR
800 19062 1B76          D0=(5) =PRGMEN         SAVE CURRST,PRGMST,PRGMEN,CURREN
          5F2
801 19069 7BF4          GOSUB  SAVE20
802               *
803 1906D 1C5           D1=D1- 6
804 19070 3500          LC(6)  #A04F00
          F40A
805 19078 15D5          DAT1=C 6
806 1907C 7CB4          GOSUB  MSTKD1        UPDATE MTHSTK BEFORE POLLING
807 19080 8E00          GOSUBL =POLL          POLL TO LET LEX FILE SAVE THEIR
          00
808 19086 73            CON(2) =pCALSV         CALL STACK
809               *
810 19088 560           GONC   CAL420        No error if carry clear
811 1908B 6A76          GOTO   bserr
812               *
813 1908F 7AC4  CAL420  GOSUB  D1FSTK
814 19093 1CF           D1=D1- 16
815 19096 1537          A=DAT1 W               RETRIEVE THE SUBPROGRAM NAME
816 1909A 102           R2=A                   R2 = SUBPROGRAM NAME
```

```
817                *
818                * See if we need to trace
819                *
820 1909D 8EE3             GOSUBL trflck          NEED TO TRACE ?
          80
821 190A3 443              GOC    CAL424
822 190A6 8F00             GOSBVL =TRCLIN
          000
823 190AD 3B02             LCASC  \ LLAC \
          3414
          C4C4
          02
824 190BB 15CB             DAT0=C 12
825 190BF 16B              D0=D0+ 12
826 190C2 11A              C=R2                    GET SUB-PROGRAM NAME
827 190C5 97A              ?C=0   W                CALL CURRENT FILE ?
828 190C8 90               GOYES  CAL422           IF SO, GOTO CAL422
829 190CA 1547             DAT0=C W
830 190CE 16F              D0=D0+ 16
831 190D1 8F00   CAL422    GOSBVL =TRTOEN
          000
832                *
833 190D8 110    CAL424    A=R0                    GET THE PC
834 190DB 130              D0=A                    D0 = PC
835 190DE 14A              A=DAT0 B                SEE IF FILE NAME SPECIFIED
836 190E1 3100             LC(2)  =tIN
837 190E5 966              ?A#C   B
838 190E8 64               GOYES  CAL425
839 190EA 161              D0=D0+ 2
840 190ED 8E00             GOSUBL =fspecx
          00
841 190F3 560              GONC   SSB20
842 190F6 6F06   SSBER     GOTO   bserr
843                *
844                *
845 190FA 8F00   SSB20     GOSBVL =FINDF+
          000
846 19101 44F              GOC    SSBER           FILE NOT FOUND
847 19104 133              AD1EX                  A= ADDRESS OF FILE START
848 19107 8E05             GOSUBL D1FSTK          RESTORE THE SUB PROGRAM NAME
          40
849 1910D 1CF              D1=D1- 16              FROM FORSTK - 16
850 19110 1577             C=DAT1 W
851 19114 10A              R2=C
852 19117 843              ST=0   3
853 1911A 855              ST=1   5               ONLY SEARCH ONE FILE
854 1911D 8E12             GOSUBL SSB210
          11
855 19123 531              GONC   CAL430          FOUND
856 19126 3100   MISUBP    LC(2)  =eSPGNF
857 1912A 6F43             GOTO   nfer1
858                *
859 1912E 8E68   CAL425     GOSUBL SSB100
          01
860 19134 41F              GOC    MISUBP
```

```
861                *
862 19137 873   CAL430 ?ST=1  3            IS THE SUB-PROG A FILE ?
863 1913A 41           GOYES  CAL450       IF SO, GOTO CAL450
864 1913C 310F         LCHEX  F0
865 19140 14A   SKNM10 A=DAT0 B
866 19143 9EE          ?A>=C  B
867 19146 80           GOYES  CAL450
868 19148 161          D0=D0+ 2
869 1914B 54F          GONC   SKNM10
870                *
871 1914E 316D  CAL450 LC(2)  =flNOFN
872 19152 8E00         GOSUBL =SFlagC      Clear no UDF flag
          00
873 19158 1F99         D1=(5) =MTHSTK
          5F2
874 1915F 143          A=DAT1 A            SET CALSTK,ACTIVE,GSBSTK FORSTK
875 19162 174          D1=D1+ 5
876 19165 2C           P=     12           AND MTHSTK TO D1
877 19167 141   CAL455 DAT1=A A
878 1916A 174          D1=D1+ 5
879 1916D 0C           P=P+1
880 1916F 57F          GONC   CAL455
881 19172 3191         LC(2)  25           LOAD CONSTANT 25
882 19176 BF0          ASL    W
883 19179 BF0          ASL    W            CLEAR ALL CHAIN HEAD COUNT AND SET
884 1917C 1FEB         D1=(5) =CHNLST       ALL CHAIN HEAD POINTERS TO CALSTK
          5F2
885 19183 1596  CAL460 DAT1=A 7
886 19187 176          D1=D1+ 7
887 1918A A6E          C=C-1  B
888 1918D 55F          GONC   CAL460
889 19190 7FEB         GOSUB  D1PRCT       D1= PRMCNT
890 19194 14F          C=DAT1 B            C= PARAMETER COUNT
891 19197 109          R1=C                SAVE PARAM COUNT IN R1 TEMP.
892 1919A 136          CD0EX                SAVE D0 IN R2
893 1919D 10A          R2=C
894 191A0 3240  CAL465 LC(3)  =bASSGN
          8
895 191A5 8E00         GOSUBL =I/OFND
          00
896 191AB 491          GOC    CAL470        ASSIGN table found
897                *
898                * Create the assign table if not yet exist
899                *
900 191AE 8E00         GOSUBL =STPASG
          00
901 191B4 8E00         GOSUBL =I/OALL
          00
902 191BA 592          GONC   noroom
903 191BD D2           C=0    A
904 191BF 145          DAT1=C A
905 191C2 4DD          GOC    CAL465        Goes around again (B.E.T.)
906                *
907 191C5 137   CAL470 CD1EX
908 191C8 C2           C=C+A  A
```

```
    909 191CA 135          D1=C                 D1 POINTS PAST END OF BUFFER
    910 191CD 1C4          D1=D1- 5
    911 191D0 147          C=DAT1 A
    912 191D3 111          A=R1                 A= PARAMETER COUNT
    913 191D6 968          ?A=0   B             PASSING ANY PARAMETER ?
    914 191D9 F0           GOYES  CAL475        IF NOT, JUST INCREASE LEVEL COUNT
    915              * PASSING PARAMETERS, GENERATE A NEW LEVEL MARKER
    916 191DB 8E00  CAL473 GOSUBL =EXPCH#
              00
    917 191E1 4C1          GOC    CAL478
    918 191E4 6D70  noroom GOTO   CAL505        NO ROOM
    919              * INCREASE LEVEL COUNT
    920 191E8 8E00  CAL475 GOSUBL =FNDMK-       LOOK FOR LOWEST LEVEL MARKER
              00
    921 191EE 171          D1=D1+ 2
    922 191F1 147          C=DAT1 A
    923 191F4 B36          C=C+1  X
    924 191F7 43E          GOC    CAL473
    925 191FA 1553         DAT1=C X
    926 191FE 11A   CAL478 C=R2                 RESTORE D0
    927 19201 134          D0=C
    928 19204 863          ?ST=0  3             SUB PROG A FILE ?
    929 19207 60           GOYES  CAL479        IF NOT, GOTO CAL480
    930 19209 66F2         GOTO   CAL830
    931 1920D 1F17  CAL479 D1=(5) =S-R0-0       MOVE S-R0-0 & S-R0-1 TO
              8F2
    932 19214 15F9         C=DAT1 10
    933 19218 1F19         D1=(5) =STMTD0        STMTD0 & STMTD1
              8F2
    934 1921F 15D9         DAT1=C 10
    935 19223 7C5B  CAL480 GOSUB  D1PRCT        GET CALL PARAMETER COUNT
    936 19227 148          A=DAT1 B
    937 1922A 3100         LC(2)  =tPRMEN       LOOK FOR PARAMETER END TOKEN
    938 1922E A6C          A=A-1  B             DECREMENT THE COUNTER
    939 19231 521          GONC   CAL500        NOT DONE WITH SUB.PARAM YET
    940 19234 14A          A=DAT0 B             SEE IF EXHAUST CALL.PARAM TOO
    941 19237 962          ?A=C   B
    942 1923A 60           GOYES  CAL495        IF SO, GOTO CAL495
    943 1923C 6332  CAL485 GOTO   CAL710
    944 19240 6D32  CAL495 GOTO   CAL800
    945              *
    946 19244 149   CAL500 DAT1=A B             UPDATE THE COUNT
    947 19247 14A          A=DAT0 B
    948 1924A 962          ?A=C   B             REACHIND END OF FORMAL PARAM LIST
    949 1924D FE           GOYES  CAL485        IF SO, PARAMETER LIST NOT MATCHING
    950 1924F 161          D0=D0+ 2             PASS OVER THE "(" OR ","
    951 19252 D2           C=0    A             SEE IF STILL ROOM ON THE STACK
    952 19254 8F00         GOSBVL =CHKSPC       CHECK IF THERE IS ENOUGH ROOM
              000
    953 1925B 501          GONC   CAL510        YES IF CARRY CLEAR
    954 1925E 7915  CAL504 GOSUB  popch#         POP OFF CHANNEL # IN LOWEST LEVEL
    955 19262 7825  CAL505 GOSUB  POPST-        POP OFF CALL STACK
    956 19266 8CD0  CALNOR GOLONG FNXNOR        SAY "INSUFFICIENT MEMORY"
              80
    957 1926C 14A   CAL510 A=DAT0 B
```

```
 958 1926F 3132          LCASC  \#\           SEE IF IS A CHANNEL #
 959 19273 966           ?A#C   B
 960 19276 73            GOYES  CAL515
 961 19278 161           D0=D0+ 2
 962 1927B 7ED2          GOSUB  D1FSTK        SET D1 TO FORSTK
 963 1927F 8EBE          GOSUBL expr          GET CHANNEL #
           D0
 964 19285 8F00          GOSBVL =POPARG       CONVERT TO HEX
           000
 965 1928C D2            C=0    A
 966 1928E A6E           C=C-1  B
 967 19291 8BA           ?A<=C  A             CHANNEL # <= 255 ?
 968 19294 01            GOYES  CAL512        IF SO, GOTO CAL512
 969 19296 71E4          GOSUB  popch#
 970 1929A 70F4          GOSUB  POPST-
 971 1929E 8C00 =InvArg  GOLONG =Invarg
           00
 972               *
 973 192A4 AF8   CAL512  B=A    W             SAVE THE CHANNEL # IN B
 974 192A7 853           ST=1   3             SET FLAG
 975 192AA 4D1           GOC    CAL518        (B.E.T.)
 976 192AD 8E00  CAL515  GOSUBL =CALFX1       GET NEXT SUB-PROG PARAMETER
           00
 977 192B3 768C          GOSUB  dest
 978 192B7 843           ST=0   3
 979 192BA D2            C=0    A             SEE IF THE VARIABLE ALREADY EXIST ?
 980 192BC B56           C=C+1  M             C(A) = 01000
 981 192BF 8B5           ?B<C   A
 982 192C2 60            GOYES  CAL518        IF SO, IS DUPLICATING PARAMETER
 983 192C4 6BA1          GOTO   CAL710        SAY PARAMETER MISMATCHING
 984 192C8 136   CAL518  CDOEX
 985 192CB 06            RSTK=C               SAVE THE PC IN RSTK
 986 192CD 7393          GOSUB  NXTPRM        GET NEXT CALL PARAMETER FROM STACK
 987 192D1 10A           R2=C                 SAVE POINTER OF ITEM IN R2
 988               *
 989               * NOW WE HAVE INFORMATION OF THE NEXT PARAMETER FROM THE CALL
 990               * LIST AND THE NEXT PARAMETER FROM THE SUB-PROG LIST
 991               * B(S) = TYPE OF THE NEXT PARAM FROM SUB-PROG
 992               *        0 - SIMPLE VARIABLE ( A )
 993               *        8 - NUMERIC ARRAY VARIABLE ( A() )
 994               *        D - STRING VARIABLE ( A$ )
 995               *        C - STRING VECTOR ( A$() )
 996               * IF P=0 PASS BY REFERENCE            IF P=1 - PASS BY VALUE
 997               * THEN:                               THEN:
 998               *  A(S)= TYPE OF NEXT PARAM FROM CALL   A(0)= TYPE OF NEXT ITEM
 999               *        0,1,2 - REAL,SHORT,INTEGER          0-9 - REAL
1000               *        8 - NUMERIC ARRAY                     E - COMPLEX
1001               *        D - STRING VARIABLE                   F - STRING
1002               *        C - STRING VECTOR
1003               *        E,F - COMPLEX,SHORT COMPLEX
1004               * A(14) = DIMENSION NUMBER IF A(S)= 8 OR C
1005               * D0 = PC
1006               * R2 POINTS AT THE ITEM OF NEXT PARAMETER IN CALL LIST, ITS
1007               *   FIRST 16 NIBBLES ARE IN THE A REGISTER
1008               *
```

```
1009 192D4 863          ?ST=0  3            IS THIS A CHANNEL # ?
1010 192D7 60           GOYES  CAL519       IF NOT, GOTO CAL519
1011 192D9 6461         GOTO   CAL660
1012 192DD 890   CAL519 ?P=    0
1013 192E0 60           GOYES  CAL520
1014 192E2 61A0         GOTO   CAL600
1015            * PASS BY REFERENCE:
1016            *
1017 192E6 8F00 CAL520  GOSBVL =CR-VAR      CREATE AN ENTRY FOR FORMAL PARAM
          000
1018 192ED 11A          C=R2                GET STACK POINTER OF ACTUAL PARAM
1019 192F0 135          D1=C
1020 192F3 1537         A=DAT1 W            GET THE ACTUAL PARAMETER ITEM TO A
1021 192F7 2F           P=     15
1022 192F9 30C          LCHEX  C
1023 192FC 942          ?A=C   S            A(S) = C ?
1024 192FF A0           GOYES  CAL530
1025 19301 A46          C=C+C  S
1026 19304 946          ?A#C   S            A(S) # 8 ?
1027 19307 93           GOYES  CAL570
1028            * ACTUAL PARAMETER IS NUMERIC ARRAY OR STRING VECTOR
1029 19309 944  CAL530  ?A#B   S            TYPE MATCH ?
1030 1930C 47           GOYES  CALERX
1031 1930E 20           P=     0
1032 19310 810          ASLC                SHIFT A(14) TO A(0)
1033 19313 810          ASLC
1034 19316 1FAB         D1=(5) =F-R1-3
          8F2
1035 1931D 1570         C=DAT1 P            READ SUBSCRIPT COUNT
1036 19321 906          ?A#C   P            IS THE SUBSCRIPT COUNT MATCH ?
1037 19324 C5           GOYES  CALERX       IF SO, ERROR OUT
1038 19326 A46   CAL550 C=C+C  S            C(S)= 0 OR 8
1039 19329 31CE         LCHEX  EC           ASSUME IS NUMERIC ARRAY
1040 1932D 94A          ?C=0   S            IF C(S)= 0, IT IS A NUMERIC ARRAY
1041 19330 50           GOYES  CAL560
1042 19332 30F          LCHEX  F
1043 19335 AEA  CAL560  A=C    B
1044 19338 1507 CAL565  DATO=A W
1045 1933C 6980         GOTO   CAL625
1046 19340 30C   CAL570 LCHEX  C            C(S) = C
1047 19343 9C6          ?A>C   S            IS A(S) > C ?
1048 19346 E1           GOYES  CAL580       IS SO, A(S) MAY BE D/E/F
1049            * ACTUAL PARAMETER IS REAL, SHORT OR INTEGER  A(S)= 0,1,2
1050 19348 94D  CAL575  ?B#0   S            THE FORMAL PARAM HAS TO BE A REAL
1051 1934B 53           GOYES  CALERX
1052 1934D D6           C=A    A            C(A)= ACTUAL PARAMETER ADDRESS
1053 1934F BF2          CSL    W
1054 19352 31CF         LCHEX  FC           C(S)= C, C(0)= F
1055 19356 B42          C=C-A  S            C(S)= C/B/A FOR REAL,SHORT,INTEGER
1056 19359 812          CSLC
1057 1935C 1547         DATO=C W            STORE THE INDIRECT ADDR TO REGISTER
1058 19360 6560         GOTO   CAL625
1059 19364 B46  CAL580  C=C+1  S            C(S)= D
1060 19367 946          ?A#C   S            IF A(S)#D IT MUST EQUAL E OR F
1061 1936A ED           GOYES  CAL575
```

```
1062 1936C 944          ?A#B   S            BOTH TYPE = D ?
1063 1936F 11           GOYES  CALERX
1064 19371 BF0          ASL    W
1065 19374 BF0          ASL    W
1066 19377 A6C          A=A-1  B
1067 1937A 4DB          GOC    CAL565         (B.E.T.)
1068 1937D 584          GONC   CAL625        RESTORE PC
1069              *
1070 19380 6FE0  CALERX GOTO   CAL710
1071              *
1072              * PASS BY VALUE
1073 19384 892   CAL600 ?P=    2            Is formal pram a channel # ?
1074 19387 9F           GOYES  CALERX        If so, error
1075 19389 20           P=     0
1076 1938B A82          C=0    P
1077 1938E A0E          C=C-1  P
1078 19391 906          ?A#C   P            IF A(0)= F IT IS A STRING
1079 19394 D3           GOYES  CAL630
1080 19396 816          CSRC
1081 19399 A4E          C=C-1  S
1082 1939C A4E          C=C-1  S            C(S)= D
1083 1939F 945          ?B#C   S            IS SUB.PARAM A STRING VARIABLE ?
1084 193A2 ED           GOYES  CALERX
1085              * STRING
1086 193A4 8F00  CAL620 GOSBVL =DFLTCR       CREATE A STRING VARIABLE
          000
1087 193AB 11A          C=R2
1088 193AE 135          D1=C
1089 193B1 2F           P=     15
1090 193B3 30D          LCHEX  D
1091 193B6 AC5          B=C    S
1092 193B9 20           P=     0
1093 193BB 1537         A=DAT1 W
1094 193BF 8F00         GOSBVL =STRASN
          000
1095 193C6 07    CAL625 C=RSTK
1096 193C8 134          DO=C
1097 193CB 20           P=     0
1098 193CD 655E         GOTO   CAL480
1099 193D1 94D   CAL630 ?B#0   S            IS THE SUB PARAMETER A SIMPLE VAR.?
1100 193D4 CA           GOYES  CALERX
1101 193D6 8F00         GOSBVL =CR-VAR       CREATE A VARIABLE OR DOPE VECTOR
          000
1102 193DD 11A          C=R2
1103 193E0 135          D1=C                 D1 POINTS TO PARAMETER VALUE
1104 193E3 30A          LCHEX  A
1105 193E6 1537         A=DAT1 W             READ THE VALUE
1106 193EA 986          ?A>C   P             IS IT A COMPLEX ?
1107 193ED 90           GOYES  CAL650        IF SO, GOTO CAL650
1108 193EF 1507  CAL640 DAT0=A W
1109 193F3 52D          GONC   CAL625        (B.E.T.)
1110 193F6 AF2   CAL650 C=0    W
1111 193F9 30E          LCHEX  E
1112 193FC 109          R1=C                 R1= DOPE VECTOR OF COMPLEX NUMBER
1113 193FF 3102         LC(2)  32            32 NIBBLES FOR A COMPLEX NUMBER
```

```
1114 19403 108          R0=C
1115 19406 8F00         GOSBVL =CR-ARR
           000
1116 1940D D4           A=B    A
1117 1940F 130          D0=A                  D0= DOPE VECTOR ADDRESS
1118 19412 16A          D0=D0+ 11
1119 19415 146          C=DAT0 A              READ RELATIVE POINTER
1120 19418 132          ADOEX
1121 1941B EA           A=A-C  A              RECTIFY ADDRESS
1122 1941D 130          D0=A
1123 19420 11A          C=R2
1124 19423 135          D1=C
1125 19426 171          D1=D1+ 2
1126 19429 1537         A=DAT1 W
1127 1942D 17F          D1=D1+ 16
1128 19430 1577         C=DAT1 W
1129 19434 1547         DAT0=C W
1130 19438 16F          D0=D0+ 16
1131 1943B 53B          GONC   CAL640         (B.E.T.)
1132              * PASSING CHANNEL #
1133              * A(2-1) = ACTUAL CHANNEL # (FROM CALL PARAM LIST)
1134              * B(B) = FORMAL CHANNEL # ( FROM SUB PARAM LIST)
1135 1943E 882   CAL660 ?P#    2              ACTUAL PARAM A CHANNEL #?
1136 19441 F2           GOYES  CAL710         IF NOT ,TYPE ERROR
1137 19443 20           P=     0
1138 19445 BB4          ASR    X              A(B) = ACTUAL CHANNEL #
1139 19448 D6           C=A    A              C(B) = ACTUAL CHANNEL #
1140 1944A F2           CSL    A
1141 1944C B06          C=C+1  P              MAKE AN INDIRECT CHANNEL #
1142 1944F F2           CSL    A
1143 19451 F2           CSL    A
1144 19453 AE9          C=B    B              C(B) = FORMAL CHANNEL #
1145 19456 10A          R2=C                  SAVE THE CHANNEL # IN R2
1146 19459 8E00         GOSUBL =EXPCH#
           00
1147 1945F 460          GOC    CAL670
1148 19462 6BFD         GOTO   CAL504         NO ROOM
1149 19466 11A   CAL670 C=R2
1150 19469 145          DAT1=C A
1151 1946C 695F         GOTO   CAL625
1152              *
1153              * REACHED END OF SUB-PROGRAM PARAMETER LIST
1154              * ABORT THE CALL IF:
1155              *   1. PARAMETER TYPE DOES'T MATCH
1156              *   2. PARAMETERS NUMBER DOES'T MATCH
1157              *   4. SAME VARIABLE NAME APPEAR MORE THAN ONCE IN SUB PARAM
1158              *      LIST.
1159              *
1160 19470 20    CAL710 P=     0
1161 19472 7413         GOSUB  POPSTK         POP OFF THE CALL STACK
1162 19476 3100 =NOTMCH LC(2)  =ePRMIS
1163 1947A 6625  nferl  GOTO   nferr
1164              *
1165              * NOW WE WANT TO EXTRACT A BLOCK OF THE STACK WE USED TO STORE
1166              * THE ACTUAL PARAMETERS (TYPE, ADDRESS OR VALUE)
```

```
1167                *
1168 1947E 1F19  CAL800 D1=(5) =STMTDO       GET LOW END ADDR OF THE GAP
           8F2
1169 19485 143          A=DAT1 A             A= LOW END ADDR OF THE GAP
1170 19488 1BE9         D0=(5) =FORSTK
           5F2
1171 1948F 146          C=DAT0 A             C= LOW END ADDR OF CURRENT STACK
1172 19492 EE           C=A-C  A             C= # OF NIBS TO PULL DOWN(MOVE UP)
1173 19494 D7           D=C    A             D=  SOURCE BLOCK SIZE
1174 19496 174          D1=D1+ 5
1175 19499 147          C=DAT1 A             C= HIGH END ADDR OF THE GAP
1176 1949C 135          D1=C                 D1= HIGH END ADDR OF DESTINATION
1177 1949F E2           C=C-A  A             C= # OF NIBBLES OF THE GAP
1178 194A1 8AA          ?C=0   A             GAP SIZE = 0 ?
1179 194A4 C5           GOYES  CAL830        IF SO DON'T DO ANYTHING
1180 194A6 FA           C=-C   A             C= 2' COMPLEMENT OF GAP SIZE
1181 194A8 DF           CDEX   A             C=SOURCE BLOCK SIZE, D= -(GAP SIZE)
1182 194AA 132          ADOEX                D0= HIGH END ADDR OF THE SOURCE
1183 194AD 7BA4         GOSUB  moved3        MOVE THE MEMORY TO CLOSE THE GAP
1184 194B1 31B5         LCHEX  5B            LOAD THE ASCII OF "Z"+1
1185 194B5 AED          BCEX   B             ADJUST ALL CHAIN HEAD ADDRESS
1186 194B8 8F00         GOSBVL =ADJPTR
           000
1187 194BF 1FDA         D1=(5) =CALSTK       ADJUST CALSTK
           5F2
1188 194C6 147          C=DAT1 A
1189 194C9 EB           C=C-D  A
1190 194CB 145          DAT1=C A
1191 194CE 135          D1=C   A
1192 194D1 7EF1         GOSUB  MANSTK        GET TO START OF MAINFRAME CALL STACK
1193 194D5 133          AD1EX
1194 194D8 D2           C=0    A              ADJUST THOSE OFFSET ON THE STACK
1195 194DA 3133         LC(2)  51
1196 194DE CA           A=A+C  A
1197 194E0 131          D1=A                  D1 POINTS TO OFFSET OF FORSTK
1198 194E3 2C           P=     12
1199 194E5 147   CAL810 C=DAT1 A
1200 194E8 CB           C=C+D  A
1201 194EA 145          DAT1=C A
1202 194ED 174          D1=D1+ 5
1203 194F0 0C           P=P+1
1204 194F2 52F          GONC   CAL810
1205 194F5 171          D1=D1+ 2             UPDATE OFFSET OF PRMPTR
1206 194F8 147          C=DAT1 A
1207 194FB CB           C=C+D  A
1208 194FD 145          DAT1=C A
1209 19500 1B26  CAL830 D0=(5) =PRGMST
           5F2
1210 19507 146          C=DAT0 A
1211 1950A 134          D0=C                   D0 POINTS TO PROG. START
1212 1950D 85D          ST=1   13              SET RUN FLAG
1213 19510 1F38         D1=(5) =ERRSUB         CLEAR ERRSUB & ERRADR
           6F2
1214 19517 AF2          C=0    W
1215 1951A 15D9         DAT1=C 10
```

```
1216 1951E 179          D1=D1+ 10            CLEAR ONINTR & DATPTR
1217 19521 15D9         DAT1=C 10
1218 19525 8EF5         GOSUBL CHKTP+        SEE IF CALLING BINARY SUBPROGRAM
          E0
1219 19528 868          ?ST=0  8
1220 1952E 60           GOYES  CAL840        IF NOT, GOTO CAL840
1221 19530 6921         GOTO   ENDS50        PASS THE EOL & JUMP TO BINARY PROG
1222              *
1223 19534 70B3  CAL840 GOSUB  SpgCsp        Set "PRGM" and clear "SUSP"
1224 19538 6CF0         GOTO   runrtn
1225              *
1226              *
1227              ********************************************************************
1228              ********************************************************************
1229              **
1230              ** Name:(S) D1MSTK  -  Set D1 at MTHSTK (AVMEME)
1231              **
1232              ** Category:   EXCUTL
1233              **
1234              ** Purpose:  Set D1 to point to available memory end (top of
1235              **           math stack)
1236              **
1237              ** Entry:  None.
1238              **
1239              ** Exit:
1240              **      D1     @  Top of math stack (available memory end)
1241              **      C(A)   =  Address of AVMEME
1242              **
1243              ** Calls:  None.
1244              **
1245              ** Uses:  C(A)
1246              **
1247              ** Stk lvls:  0
1248              ********************************************************************
1249              ********************************************************************
1250              *
1251 1953C 137  =MSTKD1 CD1EX
1252 1953F 1F99         D1=(5) =MTHSTK
          5F2
1253 19546 145          DAT1=C A
1254 19549 135          D1=C
1255 1954C 01           RTN
1256              *
1257 1954E 1F99 =D1MSTK D1=(5) =MTHSTK
          5F2
1258 19555 147   D1SUB  C=DAT1 A
1259 19558 137          CD1EX
1260 1955B 01           RTN
1261              *
1262              ********************************************************************
1263              ********************************************************************
1264              **
1265              ** Name:(S) D1FSTK  -  Set D1 to FORSTK
1266              **
1267              ** Category:   EXCUTL
```

```
1268            **
1269            ** Purpose:  Set D1 to top of FOR/NEXT stack.
1270            **
1271            ** Entry:  None
1272            **
1273            ** Exit:  D1 points at FOR/NEXT STACK
1274            **
1275            ** Uses:  C(A)
1276            **
1277            ** Stk lvls:  0
1278            ****************************************************************
1279            ****************************************************************
1280            *
1281 1955D 1FE9 =D1FSTK D1=(5) =FORSTK
           5F2
1282 19564 60FF         GOTO   D1SUB
1283            *
1284            ***************************
1285            *
1286 19568 1C9   SAVE20 D1=D1- 10
1287 1956B 15E9         C=DAT0 10
1288 1956F 15D9         DAT1=C 10
1289 19573 189   SAV10+ D0=D0- 10
1290 19576 1C9          D1=D1- 10
1291 19579 15E9  SAVE10 C=DAT0 10
1292 1957D 15D9         DAT1=C 10
1293 19581 01           RTN
1294            *
1295 19583 15F9  RSTO20 C=DAT1 10
1296 19587 15C9         DAT0=C 10
1297 1958B 169          D0=D0+ 10
1298 1958E 179          D1=D1+ 10
1299 19591 15F9  RSTO10 C=DAT1 10
1300 19595 15C9         DAT0=C 10
1301 19599 179          D1=D1+ 10
1302 1959C 01           RTN
1303            *
1304            ****************************************************************
1305            ****************************************************************
1306            ** Name:(S) ENDSUB  -  ENDSUB execution
1307            **
1308            ** Category: STEXEC
1309            **
1310            ** Purpose: End a subprogram, restore the calling program
1311            **          environment.
1312            **
1313            ** Entry: Don't care
1314            **
1315            ** Exit: Exit to NXTSTM
1316            **
1317            ** Calls: STMBUF, TRFLCK, TRCLIN, TRTOEN, POPSTK, LINSKP
1318            **        SCOPCK, CLPSTK, CLOSER, KBRTCK, SFLAGC, SFLAGS
1319            ****************************************************************
1320            ****************************************************************
1321            **
```

```
1322 1959E 0000          REL(5) =ENDSDC
           0
1323 195A3               BSS    5
1324
1325 195A8 8D00 =ENDSUB GOVLNG =END10        Do it just like the END
           000
1326 195AF 8F00 =ENDSB- GOSBVL =STMBUF
           000
1327 195B6 7723          GOSUB  trflck        NEED TO TRACE ?
1328 195BA 472           GOC    ENDS20
1329 195BD 8F00          GOSBVL =TRCLIN
           000
1330 195C4 3D02          LCASC  \BUSDNE \
           54E4
           4435
           5524
1331 195D4 15CD          DAT0=C 14
1332 195D8 16D           D0=D0+ 14
1333 195DB 8F00          GOSBVL =TRTOEN
           000
1334 195E2 74A1  ENDS20 GOSUB  POPSTK
1335 195E6 4C2           GOC    ENDS30
1336 195E9 8F00          GOSBVL =LNSKP-
           000
1337 195F0 100           R0=A                 SAVE RETURN ADDR IN R0
1338 195F3 119           C=R1                 C(0) = RETURN TYPE
1339 195F6 A06           C=C+C  P             RETURN TO BINARY PROGRAM ?
1340 195F9 406           GOC    ENDS50        IF SO, GOTO ENDS50
1341 195FC 90E           ?C#0   P             RETURN TO KEYBOARD ?
1342 195FF 92            GOYES  ENDS40        IF SO, GOTO ENDS40
1343 19601 8F00          GOSBVL =SCOPCK       CHECK IF RETURN TO CURRENT PROGRAM
           000
1344 19608 4A0           GOC    ENDS30        If lost return addr. do ENDALL
1345                *
1346 1960B 79D2  ENDS25 GOSUB  SpgCsp        Set "PRGM" and clear "SUSP"
1347 1960F 6F10          GOTO   ENDS45
1348                *
1349                * No return address can return to, do an ENDALL
1350                *
1351 19613 8F00  ENDS30 GOSBVL =CLPSTK
           000
1352 1961A 8E00          GOSUBL =CLOSEA
           00
1353 19620 3100          LC(2)  =eFnFND       IF NOT, SAY FILE NOT FOUND
1354 19624 655E          GOTO   mfer1
1355                *
1356                * KBRTCK will clear PgmRun and PGRM, CNTADR
1357                *
1358 19628 8F00  ENDS40 GOSBVL =KBRTCK       SEE IF STATEMENT EMPTY ?
           000
1359                *
1360 1962F 110   ENDS45 A=R0
1361 19632 130           D0=A
1362                *
1363 19635 3100  runrtn LC(2)  =t!
```

```
1364 19639 14A          A=DAT0 B
1365 1963C 966          ?A#C   B
1366 1963F 41           GOYES  runrt1
1367              *
1368              * Skip over comments by looking for the CR(0D)
1369              *
1370 19641 31D0         LCHEX  0D
1371 19645 161   ENDS46 D0=D0+ 2
1372 19648 14A          A=DAT0 B
1373 1964B 966          ?A#C   B
1374 1964E 7F           GOYES  ENDS46
1375 19650 161          D0=D0+ 2
1376              *
1377 19653 8D00  runrt1 GOVLNG =RUNRT1
          000
1378              *  RETURN TO BINARY PROGRAM
1379 1965A 161   ENDS50 D0=D0+ 2
1380 1965D 136          CD0EX
1381 19660 06           RSTK=C
1382 19662 03           RTNCC
1383              *
1384              **********************************
1385              *
1386              * NXTPRM - ROUTINE TO GET NEXT ITEM OF ACTUAL PARAMETER SAVED ON
1387              *          STACK.
1388              *  WHEN WE PROCESS THE ACTUAL PARAMETERS OF THE CALL STATEMENT,
1389              *  THEY ARE PROCESS ONE AT A TIME FROM LEFT TO RIGHT.  USUALLY,
1390              *  WE WRITE 16+1 NIBBLES ON THE STACK FOR THE TYPE AND ADDRESS
1391              *  OF EVERY PARAMETER. COMPLEX NUMBER USE 1+34 NIBBLES, STRING
1392              *  USE 1+16+STRING LENGTH. SO WHEN WE SCAN THROUGH THE PARAMETER
1393              *  LIST, THE LAST PARAMETER IS ON TOP OF THE STACK.
1394              *  SO WHEN WE PASS THE ADDRESS OR VALUE TO THE PARAMETER IN THE
1395              *  SUB-PROG STATEMENT, WE SHOULD GET THE NEXT ACTUAL PARAMETER
1396              *  FROM THE BOTTOM OF THE STACK.
1397              *  THE LOGIC WE USE TO GET THE NEXT ITEM FROM BOTTOM IS THIS:
1398              *  THE NEXT ITEM IS THE N.TH ITEM FROM THE TOP POINTED BY THE
1399              *  PARAMETER COUNT(PRMCNT).
1400              *
1401              *  INPUT:  (PRMCNT) = # OF REMAINING UNPROCESSED PARAMETERS
1402              *          (STMTD0) = ADDRESS OF THE TOP OF THE  PARAMETERS
1403              *                     BLOCK ON THE STACK (LOWER END ADDRESS)
1404              *  USED:  D1, P, C(A), A, D(B)
1405              *  OUTPUT:  A = THE FIRST 16 NIBBLES OF THE NEXT ITEM
1406              *           C =  POINTS TO THAT ITEM
1407              *           P= 0 - PASS BY REFERENCE
1408              *              1 - PASS BY VALUE
1409              *              2 - PASS CHANNEL #
1410              *
1411              *           CARRY SET IF R1= 0 WHEN COME IN
1412              *
1413 19664 1FB4  NXTPRM D1=(5) =PRMCNT
          9F2
1414 1966B 14F          C=DAT1 B              GET THE CURRENT PARAMETER COUNT
1415 1966E AE7          D=C    B              SAVE THE DECREMENTED COUNT IN D
1416 19671 1F19         D1=(5) =STMTD0        GET THE TOP ADDR OF THE STACK BLOCK
```

```
            8F2
1417 19678 147          C=DAT1 A
1418 1967B 135          D1=C
1419 1967E 20           P=     0
1420 19680 14F  NXPM10  C=DAT1 B            READ THE 1ST NIBS OF THE NEXT ITEM
1421 19683 170          D1=D1+ 1
1422 19686 A6F          D=D-1  B            IS THIS THE nTH ITEM WE WANT ?
1423 19689 4C3          GOC    NXPM50       YES, IF CARRY SET
1424 1968C 90E          ?C#0   P            IS ITEM PASS BY VALUE
1425 1968F 80           GOYES  NXPM20       YES, IF 1ST NIBBLE = 1
1426 19691 17F  NXPM15  D1=D1+ 16
1427 19694 5BE          GONC   NXPM10       (B.E.T.)
1428 19697 BE6   NXPM20 CSR    B            C(0) = ITEM TYPE
1429 1969A B06          C=C+1  P            IF C(0) = F, IT IS A STRING
1430 1969D 561          GONC   SKITM2       SIZE OF A STRING=1+16+STRING LENGTH
1431 196A0 171          D1=D1+ 2            POINTS TO THE STRING LENGTH
1432 196A3 147          C=DAT1 A            READ THE STRING LENGTH
1433 196A6 17D          D1=D1+ 14
1434 196A9 133          AD1EX
1435 196AC CA           A=A+C  A            POINTS TO THE END OF THE STRING
1436 196AE 131          D1=A
1437 196B1 5EC          GONC   NXPM10       (B.E.T.)
1438 196B4 B06  SKITM2  C=C+1  P            IF C(0) = E, IT IS A COMPLEX NUMBER
1439 196B7 580          GONC   SKITM5       IT IS A REAL IF NOT A COMPLEX
1440 196BA 171          D1=D1+ 2            COMPLEX NUMBER LENGTH= 1+2+16+16
1441 196BD 17F          D1=D1+ 16
1442 196C0 17F  SKITM5  D1=D1+ 16
1443 196C3 5CB          GONC   NXPM10       (B.E.T.)
1444 196C6 80D0 NXPM50  P=C    0            P = 0 OR 1 OR 2
1445 196CA 1537         A=DAT1 W            READ THE FIRST 16 NIBS OF THE ITEM
1446 196CE 137          CD1EX
1447 196D1 03           RTNCC               RETURN WITH CARRY CLEAR
1448            *
1449            ********************************************************************
1450            ********************************************************************
1451            **
1452            ** Name:    MAINSTK  -  Main stack
1453            **
1454            ** Category:   EXCUTL
1455            **
1456            ** Purpose:  Find the mainframe call stack
1457            **
1458            ** Entry:  D1 @ CALSTK
1459            **         P = 0
1460            **
1461            ** Exit:  D1 @ top of mainframe call stack
1462            **
1463            ** Calls:  None
1464            **
1465            ** Uses:  A(A), C(A), D1
1466            **
1467            ** Stk lvls:  0
1468            **
1469            ********************************************************************
1470            ********************************************************************
```

```
1471                    *
1472 196D3 143         =MANSTK A=DAT1 A            A(B)=BLOCK ID, A(4,2)=BLOCK SIZE
1473 196D6 174                 D1=D1+ 5            STEP OVER THE BLOCK HEADER
1474 196D9 3400                LC(5)  #04F00
               F40
1475 196E0 8A2                 ?A=C   A
1476 196E3 00                  RTNYES
1477 196E5 F4                  ASR    A
1478 196E7 F4                  ASR    A            A(X)=BLOCK LENG(NOT INCLUDE HEADER)
1479 196E9 137                 CD1EX
1480 196EC CA                  A=A+C  A
1481 196EE 131                 D1=A
1482 196F1 51E                 GONC   MANSTK       (B.E.T)
1483                    *
1484                    ********************************************************************
1485                    *
1486                    * RSTPTR - RESTORE SOME POINTERS OF THE CALLING PROGRAM FROM THE
1487                    *          CALL STACK
1488                    * ENTRY:  DON'T CARE
1489                    *
1490                    * EXIT:   R1= RETURN TYPE (0-PROGRAM, 1-KEYBOARD, 8-BINARY)
1491                    *         PCADDR = RETURN ADDRESS
1492                    *         CARRY CLEAR IF NO MORE SUB-PROG NESTING
1493                    *         CARRY SET IF STILL MORE SUB-PROG NESTING TO GO
1494                    *
1495                    * CALLS: SUBCHK, POLL, MANSTK, RST020, DOPCAD, RSTOPT, RSTPRM
1496                    *
1497                    * USES: A, B, C, R1, D0, D1, P
1498                    *
1499                    * STK LVL : 2
1500                    ********************************************************************
1501                    *
1502 196F4 7C60 RSTPTR  GOSUB   SUBCHK
1503 196F8 500         RTNNC
1504                    *
1505 196FB 8E00 RSTP10  GOSUBL =POLL
               00
1506 19701 63          CON(2) =pCALRS       LET LEX FILE TO RESTORE INFO.
1507 19703 580         GONC   RSTP20        No error if carry clear
1508                    *
1509 19706 8C00  bserr  GOLONG =BsErr
               00
1510                    *
1511 1970C 7450  RSTP20 GOSUB  SUBCHK
1512 19710 131         D1=A
1513 19713 7CBF        GOSUB  MANSTK       GET TO START OF MAINFRAME CALL STACK
1514                    *
1515 19717 175         D1=D1+ 5+1           D1 @ PREVIOUS PRGMST
1516 1971A 147         C=DAT1 A
1517 1971D 1C4         D1=D1- 5
1518 19720 8AA         ?C=0   A
1519 19723 00          RTNYES
1520                    *
1521 19725 1BD5        D0=(5) =CURRST       RESTORE CURRST,PRGMST,PRGMEN,CURREN
               5F2
```

```
1522 1972C 735E          GOSUB  RSTO20
1523               *
1524 19730 7D40          GOSUB  DOPCAD          RESTORE PCADDR & CNTADR
1525 19734 795E          GOSUB  RSTO10
1526               *
1527 19738 169           D0=D0+ 10             RESTORE ERRSUB - DATPTR
1528 1973B 744E          GOSUB  RSTO20
1529               *
1530 1973F 1BE9          D0=(5) =FORSTK
           5F2
1531 19746 2C            P=     12             RESTORE FORSTK,GSBSTK,ACTIVE,CALSTK
1532 19748 8E02  RSTP50  GOSUBL RSTOPT
           A0
1533 1974E 164           D0=D0+ 5
1534 19751 0C            P=P+1
1535 19753 54F           GONC   RSTP50
1536 19756 8EAF          GOSUBL RSTPRM          RESTORE PARAMETER COUNT & POINTER
           90
1537 1975C 14F           C=DAT1 B
1538 1975F 109           R1=C                   SAVE RETURN TYPE IN R1
1539 19762 03            RTNCC
1540               *
1541               *
1542 19764 1F2B =SUBCHK  D1=(5) =RAMEND
           5F2
1543 1976B 147           C=DAT1 A
1544 1976E 1C4           D1=D1- (RAMEND)-(CALSTK)
1545 19771 143           A=DAT1 A
1546 19774 8A6           ?A#C   A
1547 19777 00            RTNYES
1548 19779 03            RTNCC
1549               *
1550 1977B 8C00  popch#  GOLONG =POPCH#
           00
1551               *
1552 19781 1B97  DOPCAD  D0=(5) =PCADDR
           6F2
1553 19788 01            RTN
1554               *
1555               ********************************************************************
1556               ********************************************************************
1557               **
1558               ** Name:    RBLDCH  -  Rebuild variable chain head table
1559               **
1560               ** Category:   VARMGT
1561               **
1562               ** Purpose:  When return from a sub-program, the variable chain
1563               **    head is pointing at the local variable used by the sub-
1564               **    program. So we need to rebuild all the chain heads to
1565               **    point to the variables of the calling program.
1566               **
1567               ** Entry:  Pointer ACTIVE & CALSTK are pointing at lower and
1568               **         upper bound of the current variable set.
1569               **
1570               ** Exit:  P=0
```

```
1571                **
1572                ** Calls: ARYSIZ
1573                **
1574                ** Uses:  A,B,C,D,D0,D1,R0,R2,P
1575                **
1576                ** Stk lvls:  3
1577                ********************************************************************
1578                ********************************************************************
1579                *
1580 1978A 7DEF  POPSTK GOSUB  popch#
1581 1978E 726F  POPST- GOSUB  RSTPTR
1582 19792 400         RTNC                    RETURN IF LOST RETURN PROGRAM
1583                *
1584 19795 1FDA =RBLDCH D1=(5) =CALSTK
           5F2
1585 1979C 143         A=DAT1 A
1586 1979F 131         D1=A                    D1= CALL STACK POINTER
1587 197A2 1CF         D1=D1- 16               POINT TO 1ST VARIABLE NAME
1588 197A5 1C2         D1=D1- 3
1589 197A8 BF0         ASL    W
1590 197AB BF0         ASL    W
1591 197AE 1BEB        D0=(5) =CHNLST          POINTS TO CHAIN HEAD
           5F2
1592 197B5 20          P=     0
1593 197B7 3191        LC(2)  25
1594 197BB 1586 RBC100 DAT0=A 7
1595 197BF 166         D0=D0+ 7
1596 197C2 A6E         C=C-1  B
1597 197C5 55F         GONC   RBC100
1598 197C8 31B5        LCHEX  5B
1599 197CC AE7         D=C    B                D= LAST VARIABLE LETTER
1600 197CF 840         ST=0   0                INITIAL ARRAY EXIST FLAG
1601 197D2 133  RBC110 AD1EX
1602 197D5 1F8A        D1=(5) =ACTIVE
           5F2
1603 197DC 147         C=DAT1 A                GET THE END OF VARIABLE ADDRESS
1604 197DF 131         D1=A
1605 197E2 8BE         ?C<=A  A                PASS OVER THE END ADDR YET ?
1606 197E5 60          GOYES  RBC115
1607 197E7 64F0        GOTO   RBC240
1608 197EB 148   RBC115 A=DAT1 B               READ THE NEXT VARIABLE NAME
1609 197EE 31F5        LCHEX  5F               MASK OUT THE STRING BIT
1610 197F2 0E62        C=A&C  B
1611 197F6 963         ?C=D   B                IS NEXT VAR STILL THE SAME LETTER
1612 197F9 60          GOYES  RBC125           IF SO, GOTO RBC125
1613 197FB 6E70        GOTO   RBC200
1614                *
1615                * LOOKS LIKE THE NEXT VARIABLE STILL HAS THE SAME LETTER.
1616                * BUT NEXT TOKEN MIGHT NOT BE A VARIABLE NAME, IT COULD BE PART
1617                * OF AN ARRAY ELEMENT. SO LET'S MAKE SURE THIS IS NOT THE CASE.
1618                *
1619 197FF 860   RBC125 ?ST=0  0              ANY ARRAY IN THIS CLASS OF VARIABLE
1620 19802 E0          GOYES  RBC130           IF SO=0, THERE IS NONE. WE ARE SAFE
1621 19804 118         C=R0                    GET END OF LAST ARRAY ADDR FROM R0
1622 19807 7D46        GOSUB  A=D1             A= CURRENT VARIABLE ADDRESS
```

```
1623 1980B 8B2          ?A<C   A          ARE WE POINT INTO THE ARRAY YET ?
1624 1980E C6           GOYES  RBC200     IF SO, NEXT TOKEN NOT THE SAME LETTER
1625 19810 14E  RBC130  C=DAT0 B          INCREMENT THE HEAD COUNT
1626 19813 B66          C=C+1  B
1627 19816 14C          DAT0=C B
1628 19819 161          D0=D0+ 2            UPDATE CHAIN HEAD ADDR TOO
1629 1981C 137          CD1EX
1630 1981F 135          D1=C
1631 19822 144          DAT0=C A
1632 19825 181          D0=D0- 2
1633 19828 172          D1=D1+ 3
1634 1982B 14B          A=DAT1 B            IS THIS VAR. A POINTER REFERENCE?
1635 1982E 31A0         LCHEX  0A
1636 19832 982          ?A<C   P            IS IT A REAL NUMBER ?
1637 19835 B3           GOYES  RBC150       YES
1638 19837 E6           C=C+1  A            C(B) = 0B
1639 19839 9EA          ?A<=C  B            IS IT AN INTEGER OR SHORT ?
1640 1983C 43           GOYES  RBC150
1641 1983E 310E         LCHEX  E0
1642 19842 9EE          ?A>=C  B            MIGHT BE INDIRECT ADDRESS ?
1643 19845 B2           GOYES  RBC150       YES
1644 19847 ABF          CDEX   X            SAVE CURRENT LETTER INTO R0
1645 1984A 108          R0=C
1646 1984D 8F00         GOSBVL =ARYSIZ      COMPUTE ARRAY SIZE
          000
1647                *  NOW D1 POINTS AT THE ARRAY POINTER
1648                *      C= ARRA POINTER
1649                *      A= ARRAY SIZE
1650                *   END OF ARRAY ADDRESS = D1 - ARRAY POINTER + ARRAY SIZE
1651                *
1652 19854 EE             C=A-C  A          ARRAY SIZE - ARRAY POINTER
1653 19856 7EF5         GOSUB  A=D1
1654 1985A C2           C=A+C  A            C= END OF ARRAY ADDRESS
1655 1985C 128          CR0EX               SAVE END OF ARRAY ADDRESS IN R0
1656 1985F AB7          D=C    X            RESTORE CURRENT LETTER TO D(X)
1657 19862 870          ?ST=1  0            IS THIS THE 1ST ARRAY MET ?
1658 19865 50           GOYES  RBC140       IF NOT GOTO RBC140
1659 19867 102          R2=A                SAVE 1ST ARRAY POINTER INTO R2
1660 1986A 1CA  RBC140  D1=D1- 11           POINTS BACK TO START OF DOPE VECTOR
1661 1986D 850          ST=1   0            SET ARRAY FLAG
1662 19870 1C5  RBC150  D1=D1- 6            POINTS TO NEXT VARIABLE NAME
1663 19873 1CF  RBC155  D1=D1- 16
1664 19876 6B5F RBC160  GOTO   RBC110
1665                *
1666                *  REACHED END OF A VARIABLE CLASS
1667                *
1668 1987A 860  RBC200  ?ST=0  0            ANY POINTER REF. IN THIS CLASS?
1669 1987D C1           GOYES  RBC210       IF NOT GOTO RBC210
1670 1987F 12A          CR2EX               GET 1ST ARRAY POINTER FROM R2
1671 19882 135          D1=C
1672 19885 147          C=DAT1 A            READ THE ARRAY POINTER
1673 19888 133          AD1EX               A= ADDRESS OF ARRAY POINTER
1674 1988B EA           A=A-C  A            A= START OF THE ARRAY
1675 1988D 131          D1=A
1676 19890 1C2          D1=D1- 3
```

```
1677 19893 840          ST=0   0
1678 19896 5CD          GONC   RBC155        (B.E.T.)
1679 19899 14B  RBC210  A=DAT1 B             READ THE NEW LETTER AGAIN
1680 1989C 31F5         LCHEX  5F            MASK OFF STRING & TRACE BITS
1681 198A0 0E66         A=A&C  B
1682 198A4 3114 RBC220  LCASC  \A\
1683 198A8 A6F          D=D-1  B             POINTS TO NEXT LETTER IN THE CHAIN
1684 198AB 186          DO=DO- 7             POINTS TO NEXT CHAIN HEAD ADDR
1685 198AE 9EB          ?D>=C  B             SHOT OVER LETTER "A" YET ?
1686 198B1 40           GOYES  RBC230
1687 198B3 03           RTNCC                IF SO, WE ARE DONE
1688 198B5 161   RBC230 DO=DO+ 2               MOVE UP CHAIN HEAD ADDRESS FIRST
1689 198B8 17F          D1=D1+ 16
1690 198BB 172          D1=D1+ 3
1691 198BE 137          CD1EX
1692 198C1 144          DAT0=C A
1693 198C4 137          CD1EX
1694 198C7 1CF          D1=D1- 16
1695 198CA 1C2          D1=D1- 3
1696 198CD 181          DO=DO- 2
1697 198D0 AEB          C=D    B
1698 198D3 966          ?A#C   B             IS THIS THE NEXT LETTER ?
1699 198D6 EC           GOYES  RBC220        IF NOT, SKIP THIS LETTER
1700 198D8 662F         GOTO   RBC125
1701               *
1702 198DC D0    RBC240 A=0    A
1703 198DE 55C          GONC   RBC220        (B.E.T.)
1704               *
1705 198E1 8D00  trflck GOVLNG =TRFLCK
           000
1706               *
1707               *
1708               * SpgCsp - Set "PRGM" and clear "SUSP" annociator
1709               *
1710 198E8 8D00  SpgCsp GOVLNG =SFGPGM
           000
1711               *
1712               ************************************************************
1713               ************************************************************
1714               ** Name:    DEF     -  Execution of DEF statement.
1715               **
1716               ** Category: STEXEC
1717               **
1718               ** Purpose: When program execution encounter the DEF statement,
1719               **          skip over the entire user-define function block.
1720               **
1721               ** Entry: DO pts past the DEF token
1722               **
1723               ** Exit: Exit to NXTSTM
1724               **
1725               ** Uses: A(A), B(A), C(A), DO
1726               **
1727               ** Stk lvl: 0
1728               **
1729               ** Detail:
```

```
1730                ** What follows the DEF token is the Label chain and after
1731                ** the Label chain is a token will indicate this is a single
1732                ** line or multi-line UDF. If the token is a non-zero byte,
1733                ** this is a single line UDF, else this is a multi-line UDF.
1734                ** The way to skip over a multi-line UDF is to look for the
1735                ** ENDDEF token. But if we run into another DEF token before
1736                ** we find the ENDDEF token, then we know the UDFs are
1737                ** nested.
1738                **
1739                ********************************************************************
1740                ********************************************************************
1741                **
1742 198EF 0000          REL(5) =DEFDC
          0
1743 198F4 0000          REL(5) =DEFP
          0
1744 198F9 164  =DEF     D0=D0+ 5            D0 PASSED THE ADDREESS FIELD
1745 198FC 14A           A=DAT0 B            READ THE tFN TOKEN
1746 198FF 96C           ?A#0   B            IS THIS A SINGLE LINE FUNCTION ?
1747 19902 25            GOYES  DEF60        IF SO, SKIP OVER THIS LINE
1748 19904 D1            B=0    A            PREPARE FOR NESTING DEF STRUCTURE
1749 19906 184           D0=D0- 5            POINTS BACK TO ADDRESS FIELD
1750 19909 146   DEF10   C=DAT0 A            READ THE LINKT LINK
1751 1990C E6            C=C+1  A            ARE WE AT END OF CHAIN ?
1752 1990E 494           GOC    DEF70        IF SO, do an END
1753 19911 7B45          GOSUB  D0+C
1754 19915 182           D0=D0- 3            POINTS TO NEXT LINK -2
1755 19918 3100          LC(2)  =tENDDF      SEE IT IS AN "ENDDEF" TOKEN
1756 1991C 14A           A=DAT0 B
1757 1991F 161           D0=D0+ 2            POINTS BACK TO LINK FIELD
1758 19922 962           ?A=C   B            IS THIS A ENDDEF TOKEN ?
1759 19925 F1            GOYES  DEF30
1760 19927 3100          LC(2)  =tDEF
1761 1992B 966           ?A#C   B            IS THIS ANOTHER DEF ?
1762 1992E BD            GOYES  DEF10        IF NOT GOTO DEF20
1763 19930 164           D0=D0+ 5
1764 19933 14A           A=DAT0 B            READ THE tFN TOKEN
1765 19936 184           D0=D0- 5            POINTS BACK TO LINK FIELD
1766 19939 96C           ?A#0   B            SINGLE LINE FUNCTION ?
1767 1993C DC            GOYES  DEF10        IF SO, GOTO DEF10
1768 1993E B65           B=B+1  B
1769 19941 57C           GONC   DEF10
1770 19944 A6D   DEF30   B=B-1  B            END OF DEF NESTING YET ?
1771 19947 51C           GONC   DEF10        IF NOT, KEEP LOOKING
1772 1994A 183           D0=D0- 4            D0 POINTS TO LINE LEN OF ENDDEF
1773                * HAVE NOT TOUCHED sENDx
1774 1994D 8D00          GOVLNG =NXTST2
          000
1775 19954 62B0  DEF60   GOTO   nxtstm
1776                *
1777 19958 6F4C  DEF70   GOTO   ENDSUB
1778                *
1779                *
1780 1995C 8C00  moved3  GOLONG =MOVED3
          00
```

```
1781                    *
1782                    ********************************************************************
1783                    ********************************************************************
1784                    **
1785                    ** Name:    FCNADR  -  Function address
1786                    **
1787                    ** Category:   VARMGT
1788                    **
1789                    ** Purpose:  Search the current function value on the stack
1790                    **           This is for the function assignment statement
1791                    **
1792                    ** Entry:  D0= PC pts past the function name
1793                    **         B(X) = Function name (generated by the ADRSUB)
1794                    **
1795                    ** Exit:  A = PC (input of D0)
1796                    **        D0 = Address to store the function value
1797                    **        B = D0
1798                    **        If function not found, exit to error routine
1799                    ** Calls:  D1FSTK
1800                    **
1801                    ** Uses:  A(A),B(A),C(A),D(B),D0,D1
1802                    **
1803                    ** Stk lvls: 1
1804                    ********************************************************************
1805                    ********************************************************************
1806                    *
1807 19962 77FB  FCNADR GOSUB  D1FSTK        SET D1 TO FORSTK
1808 19966 132         ADOEX                LOOK FOR THE FUNCTION NAME
1809 19969 1B7B        D0=(5) =PRMPTR
           5F2
1810 19970 14E         C=DAT0 B
1811 19973 A6E         C=C-1  B
1812 19976 462         GOC    FNMISS        NOT IN DEF FN IF PARAM COUNT =0
1813 19979 AE7         D=C    B
1814 1997C 161         D0=D0+ 2
1815 1997F 146         C=DAT0 A
1816 19982 134         D0=C
1817 19985 A6F   FNX110 D=D-1  B            GET TO FUNCTION NAME
1818 19988 4B0         GOC    FNX120
1819 1998B 162         D0=D0+ 3
1820 1998E 16F         D0=D0+ 16
1821 19991 53F         GONC   FNX110        (B.E.T.)
1822 19994 1563  FNX120 C=DAT0 X            READ CURRENT FUNCTION NAME
1823 19998 931         ?B=C   X
1824 1999B C0          GOYES  FNX130        THIS IS THE RIGHT FUNCTION NAME
1825 1999D 3100 =FNMISS LC(2)  =eFNNtF
1826 199A1 8C00  nferr  GOLONG =MFERRj
           00
1827 199A7 8D00  FNX130 GOVLNG =ADRS80       DO THE FOLLOWING INSTRUCTIONS
           000
1828                    *
1829                    *       D0=D0+ 3     PASS FN NAME
1830                    *       ADOEX        SAVE ADDR TO B(A)
1831                    *       B=A    A
1832                    *       ADOEX
```

```
1833            *       RTNCC
1834            *
1835            ****************************************************************
1836            ****************************************************************
1837            **
1838            ** Name:    FN      -  Function execution
1839            ** Name:    FNBIN   -  Function execution
1840            **
1841            ** Category:   STEXEC
1842            **
1843            ** Purpose :
1844            **       1. Evalue user defineded function within an expression
1845            **          execution.
1846            **       2. Execution of a FN assignment statement.
1847            **
1848            ** Entry:  D0 pts at the function name (past the tFN token)
1849            **         D1 pts top of the math stack
1850            **
1851            ** Exit:
1852            **        This routine return to EXPR, so can't be called as
1853            **        a routine.
1854            **        D0 pts past the function token.
1855            **
1856            **
1857            ** Calls: FCNADR, GETNAM, PTRRCL, RECALL, ASNSTO, CR-PAR
1858            **        EXPEXC, FLTDH, STKCHK, MOVEU3, MOVED3, TRFLCK, TRFROM
1859            **        COMPL#, TRTO*, SFLAG?, PRSCKB, FNDFCN, PTRSUB, GETPC
1860            **        FCALC?, SAVPTR, DOPCAD, FPOLL
1861            **
1862            ** Uses:  Everything
1863            **
1864            ** Stk lvls:  +6
1865            **
1866            ** Detail:
1867            **
1868            ** 1. The formal parameters of a function are local variables to
1869            **    the function.
1870            **
1871            ** 2. The local variables of a function are stored on the math
1872            **    stack on top of the current math stack.
1873            **
1874            ** 3. PRMPTR are a seven nibbles vector which point to the
1875            **    function parameter stack. The first two nibbles is the
1876            **    parameter count. The count is equal to the number of
1877            **    parameters minus one.
1878            **
1879            **    Parameter count = 0 - Not executing an UDF.
1880            **          //        = 1 - The UDF has no parameter.
1881            **          //        = n - Number of parameters equal to n-1.
1882            **
1883            ** 4. If parameter count is non-zero, the next five nibbles is
1884            **    the address of the first parameter. All the dope vectors
1885            **    of the parameter are next to each other just like a block
1886            **    of variables that have the same letter.
1887            **
```

```
1888            ** 5. The dope vector of the function value is placed after the
1889            **    parameters just above the old math stack.
1890            **
1891            ** 6. Above the parameter stack, there is a block of memory used
1892            **    as a function stack.
1893            **    The items stacked by an user defineded function are:
1894            **
1895            **  a. F00000 (6) - This five zeros prevent the function stack
1896            **                  been taking as GOSUB return address.
1897            **
1898            **  b. program pointer (5)   Following 7 pointers are absolute
1899            **  c. PCADDR          (5)   addresses, they will be justified
1900            **     CNTADR          (5)   when program memory moved.
1901            **  d. STMTD0          (5)
1902            **
1903            **  e. 3 addresses poped off from the hardware stack (15)
1904            **
1905            **  f. STMTD1 (5)  This five nibbles are assumed to be stack
1906            **                 pointer. If the user function creates a new
1907            **                 variable, this address will be justified.
1908            **
1909            **  g. STMTR0 (16)  Following 32 nibbles are statement scratch
1910            **  h. STMTR1 (16)  register 0 & 1. STMTR0[4:0] is assumed to be
1911            **                  destination of the assignment statement. It
1912            **                  will be justified if math stack moved.
1913            **                  STMTR1[14:10] is assumed to be the PC points
1914            **                  at the destination variable. It will be
1915            **                  justified if program memory moved.
1916            **
1917            **  i. Offset to previous MTHSTK (5)
1918            **  j. Offset to previous FORSTK (5)
1919            **  k. Offset to previous GSBSTK (5)
1920            **  l. Previous parameter count  (2)
1921            **  m. Offset to previous PRMPTR+2 (5)
1922            **  n. STSAVE (3)
1923            **  o. CHN#SV (2)
1924            **  p. Return type (1)  0 - Executed from program execution
1925            **                      1 - Executed from keyboard
1926            **                      8 - Called by a binary program
1927            **
1928            ********************************************************************
1929            ********************************************************************
1930            *
1931            *
1932 199AE CCCC         NIBHEX CCCCCCCCCCCCC
           CCCC
           CCCC
           CC
1933 199BC 0E           NIBHEX 0E              ARGUMENT RANGE (0,14)
1934 199BE 132  =FN     ADOEX                  Save PC in A temp.
1935 199C1 130          DO=A
1936 199C4 7DB6         GOSUB  GETNAM          GET THE FUNCTION NAME
1937 199C8 AE2          C=0    B
1938 199CB 1560         C=DAT0 P               READ THE PARAMETER COUNT
1939 199CF B06          C=C+1  P               IS THIS A FUNCTION ASSIGNMENT ?
```

```
1940 199D2 5A3          GONC   FNX200        NO, IF PARAMETER COUNT # F
1941               *
1942               * Save the PC in S-R1-2 for trace
1943               *
1944 199D5 1FB8         D1=(5) =S-R1-2
           8F2
1945 199DC CC           A=A-1  A             Points back to tFN
1946 199DE CC           A=A-1  A
1947 199E0 141          DAT1=A A
1948               *
1949 199E3 7B7F         GOSUB  FCNADR
1950 199E7 E4           A=A+1  A             PAST THE PARAMETER COUNT
1951 199E9 860          ?ST=0  0             IS THIS NUMERIC FUNCTION ?
1952 199EC D0           GOYES  FNX140        IF SO GOTO FNX140
1953 199EE 8F00         GOSBVL =PTRRCL
           000
1954 199F5 6A00         GOTO   FNX150
1955 199F9 8F00  FNX140 GOSBVL =RECALL
           000
1956               *
1957 19A00 8F00  FNX150 GOSBVL =ASNSTO
           000
1958               *
1959 19A07 8C00  nxtstm GOLONG =NXtstm       GOTO EXECUTE NEXT STATEMENT
           00
1960               *
1961 19A0D A0E   FNX200 C=C-1  P             C(0) = PARAMETER COUNT
1962 19A10 160          D0=D0+ 1             PASS PARAMETER COUNT
1963 19A13 132          AD0EX                A=PC
1964 19A16 1BAA         D0=(5) =F-R0-3
           8F2
1965 19A1D 108          R0=C                 R0= PARAMETER COUNT
1966 19A20 1540         DAT0=C P             F-R0-3 = PARAMETER COUNT
1967 19A24 184          D0=D0- 5
1968 19A27 140          DAT0=A A             F-R0-2 = SAVED PC
1969 19A2A 184          D0=D0- 5
1970 19A2D 133          AD1EX
1971 19A30 140          DAT0=A A             F-R0-1 = PARAM VALUE STACK POINTER
1972 19A33 D4           A=B    A
1973 19A35 101          R1=A                 R1 = FUNCTION NAME
1974               *
1975               * Test if flNOFN flag set ?
1976               *
1977 19A38 316D         LC(2)  =flNOFN       Is UDF disallowed ?
1978 19A3C 8E00         GOSUBL =SFLAG?
           00
1979 19A42 5A0          GONC   FNX206
1980               *
1981 19A45 3100         LC(2)  =eILPAR
1982 19A49 675F         GOTO   mferr         If so, say "Illegal Parm"
1983               *
1984               * SEARCH THE DEF FN FROM START OF THE FILE
1985               *
1986 19A4D 8F00  FNX206 GOSBVL =PRSCKB       Set program scope first
           000
```

```
1987 19A54 7056          GOSUB  FNDFN-
1988 19A58 460           GOC    FNX240
1989 19A5B 614F          GOTO   FNMISS
1990 19A5F 1F0A  FNX240  D1=(5) =F-R0-1
          8F2
1991 19A66 147           C=DAT1 A
1992 19A69 135           D1=C                 D1 POINTS TO 1ST PARAMETER VALUE
1993 19A6C 8E00          GOSUBL =STK19?       ROOM FOR 19 NIBBLES ON STACK ?
          00
1994 19A72 480           GOC    FNX260        YES IF CARRY SET
1995 19A75 8C00  FNXNOR  GOLONG =MEMERJ       GOTO MEMERR
          00
1996 19A7B 3108  FNX260  LCHEX  80
1997 19A7F 0E69          B=B!C  B             B(X)= FUNCTION NAME
1998 19A83 7386          GOSUB  CR-PAR
1999 19A87 860           ?ST=0  0             IS THIS NUMERIC FUNCTION ?
2000 19A8A 04            GOYES  FNX280        IF SO, GOTO FNX280
2001 19A8C 3100          LC(2)  =tSEMIC       SEE IF STRING LENGTH SPECIFIED
2002 19A90 14A           A=DATO B
2003 19A93 966           ?A#C   B
2004 19A96 43            GOYES  FNX280        IF NOT,  USED DEFAULT STRING LENGTH
2005 19A98 161           D0=D0+ 2             PASS THE SEMICOLON
2006 19A9B 71D5          GOSUB  expr
2007 19A9F 161           D0=D0+ 2             PAST TAIL tSEMIC
2008 19AA2 17F           D1=D1+ 16
2009 19AA5 8E00          GOSUBL =FLTDH        CONVERT FLOATING NUMBER TO BINARY
          00
2010 19AAB 480           GOC    FNX270        IF < 65535, USE IT AS SPECIFIED
2011               *
2012 19AAE 8CEE  FNX265  GOLONG InvArg        OTHERWISE, SAY "Invalid Arg."
          7F
2013               *
2014 19AB4 24    FNX270  P=     4             SEE IF THE NUMBER IS > 65535
2015 19AB6 90C           ?A#0   P
2016 19AB9 20            GOYES  FNX275
2017 19ABB 20    FNX275  P=     0
2018 19ABD 40F           GOC    FNX265
2019 19AC0 175           D1=D1+ 6             POINTS TO MAX. LENGTH
2020 19AC3 1593          DAT1=A 4
2021 19AC7 1C5           D1=D1- 6
2022 19ACA 160   FNX280  D0=D0+ 1
2023 19ACD 3100          LC(2)  =tPRMST       SEE IF THIS FN TAKES ANY PARAMETER
2024 19AD1 14A           A=DATO B             READ TOKEN FOLLOWS FN NAME
2025 19AD4 966           ?A#C   B             IS AN "(" FOLLOWS ?
2026 19AD7 52            GOYES  FNX310        IF NO PARAM REQUIRED, GOTO FNX310
2027 19AD9 161           D0=D0+ 2             PASS THE "("
2028 19ADC 75A5  FNX300  GOSUB  GETNAM        GET THE 1ST PARAMETER NAME
2029 19AE0 8E00          GOSUBL =STK19?
          00
2030 19AE6 5E8           GONC   FNXNOR
2031 19AE9 7D16          GOSUB  CR-PAR        CREATE DOPE VECTOR FOR THE PARAM
2032 19AED 3192          LCASC  \)\
2033 19AF1 14A           A=DATO B
2034 19AF4 966           ?A#C   B             REACH THE ")" YET ?
2035 19AF7 5E            GOYES  FNX300        IF NOT GOTO FNX300, MORE PARAMETERS
```

```
2036 19AF9 161          DO=DO+ 2            PASS THE ")"
2037 19AFC 345B FNX310  LC(5)  =F-R1-2
           8F2
2038 19B03 136          CDOEX
2039 19B06 144          DAT0=C A            F-R1-2 = ADDR OF END OF DEF FN
2040 19B09 184          DO=DO- 5
2041 19B0C 137          CD1EX
2042 19B0F 144          DAT0=C A            F-R1-1 = WILL BE THE CURRENT PRMPTR
2043 19B12 10A          R2=C                R2 = CURRENT STACK POINTER
2044 19B15 18F          DO=DO- 16
2045 19B18 142          A=DAT0 A
2046 19B1B 131          D1=A                D1 PTS TO 1ST PARAM VALUE ON STACK
2047 19B1E 134          DO=C                DO POINTS TO LAST PARAMETER
2048             * NOW R2 = RUNNING CURRENT STACK POINTER
2049             *     D1 = POINTS TO NEXT PARAMETER VALUE ON THE STACK
2050             *     DO = POINTS TO NEXT PARAMETER NAME
2051             *     R0 = REMAINING PARAMETER COUNT
2052             * ASSIGN THE VALUE TO THE PARAMETERS
2053             *
2054 19B21 840          ST=0   0            RESET COMPLEX NUMBER FLAG
2055 19B24 1563 FNX320 C=DAT0 X            READ NEXT PARAMETER NAME
2056 19B28 A66          C=C+C  B            IS THIS THE FUNCTION NAME ?
2057 19B2B 560          GONC   FNX330       IF NOT GOTO FNX330
2058 19B2E 6FC0         GOTO   FNX400       FN VAR IS LAST ON PARAMETER CHAIN
2059 19B32 162  FNX330  DO=DO+ 3
2060 19B35 128          CR0EX               C= REMAINING PARAMETER COUNT
2061 19B38 A0E          C=C-1  P            IF # OF ACTUAL PARAMETERS LESS THAN
2062 19B3B 4A2          GOC    FNXTER        REQUIRED BY DEF FN, IS AN ERROR
2063 19B3E 128          CR0EX
2064 19B41 1537         A=DAT1 W            READ THE PARAMETER VALUE
2065 19B45 30A          LCHEX  A
2066 19B48 982          ?A<C   P            IS THE VALUE A REAL NUMBER ?
2067 19B4B F1           GOYES  RVALUE       IF SO, GOTO RVAULE
2068 19B4D 171          D1=D1+ 2
2069 19B50 31E0         LCHEX  0E
2070 19B54 962          ?A=C   B            IS THE VALUE A COMPLEX NUMBER ?
2071 19B57 A2           GOYES  CVALUE       IF SO, GOTO CVALUE
2072 19B59 D6           C=A    A
2073 19B5B B06          C=C+1  P            IS THE VALUE A STRING ?
2074 19B5E A66          C=C+C  B
2075 19B61 96A          ?C=0   B            IS THIS A STRING VALUE ?
2076 19B64 35           GOYES  SVALUE       IF SO, OK
2077 19B66 6F09 FNXTER  GOTO   NOTMCH       SAY "PARAMETER NOT MATCH"
2078 19B6A 1563 RVALUE  C=DAT0 X
2079 19B6E 93E          ?C#0   X            IS THE FORMAL PARAM A REAL TYPE ?
2080 19B71 5F           GOYES  FNXTER       IF NOT, GOTO FNXTER
2081 19B73 1507         DAT0=A W            ASSIGN THE VALUE TO THE PARAMETER
2082 19B77 17F          D1=D1+ 16           POINTS TO NEXT PARAMETER NAME
2083 19B7A 16F          DO=DO+ 16           POINTS TO NEXT PARAMETER VALUE
2084 19B7D 66AF         GOTO   FNX320
2085 19B81 850  CVALUE  ST=1   0            REMEMBER COMPLEX IN ACTUAL PARAM
2086 19B84 146          C=DAT0 A
2087 19B87 93E          ?C#0   X            IS FORMAL PARAMETER A REAL TYPE ?
2088 19B8A CD           GOYES  FNXTER       IF NOT, GOTO FNXTER
2089 19B8C 148          DAT0=A B            MAKE THE DOPE VECTOR A COMPLEX TYPE
```

```
2090 19B8F 3102          LC(2)  32            32 NIBS REQUIRED FOR COMPLEX NUMBER
2091 19B93 78E5          GOSUB  PTRSUB        CREATE DOPE VECTOR & ALLOCATE SPACE
2092               *
2093               * Reverse the order of imaginary and real part of the complex
2094               * number.
2095               *
2096 19B97 1567          C=DATO W
2097 19B9B 16F           DO=DO+ 16
2098 19B9E 1527          A=DATO W
2099 19BA2 1547          DATO=C W
2100 19BA6 18F           DO=DO- 16
2101 19BA9 1507          DATO=A W
2102               *
2103 19BAD D2            C=0    A
2104 19BAF 3102          LC(2)  32
2105 19BB3 6830          GOTO   FNX380
2106 19BB7 146    SVALUE C=DATO A
2107 19BBA 906           ?A#C   P             IS THE PARAMETER A STRING TYPE
2108 19BBD 9A            GOYES  FNXTER        IF NOT, GOTO FNXTER
2109 19BBF D2            C=0    A
2110 19BC1 304           LC(1)  4             ADD 4 NIBBLES TO THE STRING LENGTH
2111 19BC4 143           A=DAT1 A             READ THE STRING LENGTH
2112 19BC7 C2            C=A+C  A             C= REQUIRED SPACE
2113 19BC9 162           DO=DO+ 3             WRITE THE MAX. LENG TO DOPE VECTOR
2114 19BCC 81C           ASRB                 A= STRING LENGTH IN BYTES
2115 19BCF 1583          DATO=A 4
2116 19BD3 182           DO=DO- 3
2117 19BD6 75A5          GOSUB  PTRSUB        CREATE DOPE VECTOR & ALLOCATE SPACE
2118               *
2119               * NOW B(A) = POINTS TO NEXT PARAMETER
2120               *     R2 & D1 = CURRENT STACK POINTER
2121               *     DO = POINTS TO NEXT PARAMETER VALUE
2122               *
2123 19BDA 142           A=DATO A             READ THE STRING LENGTH
2124 19BDD D6            C=A    A             C= STRING LENGTH IN NIBBLES
2125 19BDF 16D           DO=DO+ 14
2126 19BE2 81C           ASRB                 A= STRING LENGTH IN BYTES
2127 19BE5 1593          DAT1=A 4             WRITE THE LENGTH TO STRING HEAD
2128 19BE9 173           D1=D1+ 4
2129 19BEC 8E00   FNX380 GOSUBL =MOVEU3       MOVE STRING OR COMPLEX NUMBER
          00
2130 19BF2 D9            C=B    A
2131 19BF4 136           CDOEX                DO POINTS TO NEXT PARAMETER NAME
2132 19BF7 135           D1=C                 D1 POINTS TO NEXT PARAMETER VALUE
2133 19BFA 692F          GOTO   FNX320
2134               * WE HAVE ASSIGN VALUE TO ALL THE PARAMETERS . NOW WE WILL
2135               * ALLOCATE SPACE TO THE FUNCTION VARIABLE IF IT IS A STRING FN.
2136               * IF A NUMERIC FUNCTION WITH ANY OF ITS PARAMETER COME IN AS A
2137               * COMPLEX NUMBER, THE FUNCTION WILL BECOME A COMPLEX FUNCTION
2138 19BFE 14A    FNX400 A=DATO B             MASK OFF THE BIT SET FOR FN NAME
2139 19C01 31F7          LCHEX  7F
2140 19C05 0E66          A=A&C  B
2141 19C09 148           DATO=A B
2142 19C0C 162           DO=DO+ 3
2143 19C0F 146           C=DATO A
```

```
2144 19C12 93E          ?C#0   X            IS THIS A STRING FUNCTION ?
2145 19C15 C1           GOYES  FNX420       IF SO, GOTO FNX420
2146 19C17 870          ?ST=1  0            ANY COMPLEX NUMBERIN PARAMETER ?
2147 19C1A 80           GOYES  FNX410       IF SO GOTO FNX410
2148 19C1C 16F          D0=D0+ 16           PASS THE FUNCTION VALUE
2149 19C1F 564          GONC   FNX450       UNCONDITIONAL GOTO FNX450
2150 19C22 31E0 FNX410  LCHEX  0E           GENERATE COMPLEX DOPE VECTOR
2151 19C26 14C          DAT0=C B
2152 19C29 3102         LC(2)  32           ALLOCATE 32 NIBBLES FOR COMPLEX FN
2153 19C2D 6510         GOTO   FNX430
2154 19C31 162  FNX420  D0=D0+ 3            PTS TO MAX. STRING LENGTH OF FNF$
2155 19C34 D2           C=0    A
2156 19C36 15E3         C=DAT0 4            C=STRING FCN LENGTH IN BYTES
2157 19C3A E6           C=C+1  A            ADD TWO BYTES FOR STRING HEADER
2158 19C3C E6           C=C+1  A
2159 19C3E C6           C=C+C  A            MULTIPLY BY 2 = # OF NIBS
2160 19C40 182          D0=D0- 3
2161 19C43 7835 FNX430  GOSUB  PTRSUB
2162 19C47 AF2          C=0    W            ZERO THE NUMBER OR STRING
2163 19C4A 15D3         DAT1=C 4
2164 19C4E 860          ?ST=0  0            NOT A COMPLEX FUNCTION ?
2165 19C51 D0           GOYES  FNX435       IF NOT, ONLY ZERO STRING LENG.
2166            *
2167 19C53 1557         DAT1=C W
2168 19C57 17F          D1=D1+ 16
2169 19C5A 1557         DAT1=C W
2170 19C5E D9   FNX435  C=B    A
2171 19C60 136          CDOEX               D0 POINTS TO START OF OLD STACK
2172 19C63 135          D1=C                D1 POINTS TO END OF PARAMETER STACK
2173            * NOW D1 = STACK POINTER PASS ALL THE PARAMETER VALUES
2174            *     D0 = STACK POINTER OF TOP OF THE PARAMETER VALUES
2175            *     R2 = CURRENT TOP OF STACK POINTER
2176            *     R0 = # OF PARAMETER'S VALUES ON STACK THAT IS NOT REQUIRED
2177 19C66 128  FNX450 CROEX
2178 19C69 A0E          C=C-1  P            # OF ACTUAL AND FORMAL PARAM MATCH?
2179 19C6C 460          GOC    FNX490       IF SO, GOTO FNX490
2180 19C6F 66FE         GOTO   FNXTER       SAT "PARAMETER MISMATCH"
2181            * NOW WE WANT TO EXTRACT THE PARAMETER VAULE FROM THE STACK
2182 19C73 122  FNX490  AR2EX               A= CURRENT TOP OF STACK POINTER
2183 19C76 137          CD1EX
2184 19C79 135          D1=C
2185 19C7C D7           D=C    A            D= BOTTOM OF VALUE STACK ADDRESS
2186 19C7E 136          CDOEX               C= STK.PTR OF TOP OF PARAM VALUE
2187 19C81 134          D0=C
2188 19C84 E3           D=D-C  A            D= # OF NIBBLES TO BE EXTRACTED
2189 19C86 E2           C=C-A  A            C= STACK SIZE TO BE MOVED
2190 19C88 70DC         GOSUB  moved3
2191            *
2192            * F-R0-1 = STACK POINTER WHEN THE FUNCTION IS CALLED
2193            * F-R0-2 = PC PAST THE PARAMETER COUNT OF THE CALLING ROUTINE
2194            *          ( RETURN ADDRESS)
2195            * F-R0-3 = PARAMETERS COUNT
2196            * F-R1-0 = ADDRESS PAST THE tDEF TOKEN OF THE DEF FN
2197            * F-R1-1 = PARAMETERS CHAIN HEAD OF THE FUNCTION'S VARIABLES
2198            * F-R1-2 = PC OF THE END OF THE DEF FN STATEMENT(START EXECUTION
```

```
2199            *            ADDRESS OF THE FUNCTION)
2200            *
2201            ************************************
2202            *
2203            * NOW D1= NEW TOP OF STACK POINTER
2204            * F-R1-1 = NEW "PRMPTR" SHOULE BE BEFORE WE EXTRACT THE PARAM
2205            *          VALUES ( NOW IS OFFSET BY D(A))
2206            * WE WILL STORE THE FOLLOWING POINTERS ON STACK:
2207            *   1. F00000 (6) - THIS 5 NIBS OF ZERO IS TO PREVENT
2208            *                   "RETURN" TO UES GARBAGE FOR RETURN ADDRESS
2209            *   2. PROGRAM POINTER (5) - WILL BE UPDATED WHEN MEMORY MOVE
2210            *   3. PCADDR (5) - WILL BE UPDATED WHEN MEMORY MOVE
2211            *      CNTADR (5) -            //
2212            *   4. STMTDO (5) - WILL BE UPDATED WHEN MEMORY MOVE
2213            *   5. 3 HARDWARE RETURN ADDRESS (15)
2214            *   6. STMTD1 (5) - WILL BE UPDATED WHEN STACK MOVE
2215            *   7. S-R0-0,1,2,3 (16)
2216            *   8. S-R1-0,1,2,3 (16)
2217            *   9. OFFSET TO PREVIOUS MTHSTK (5)
2218            *  10. OFFSET TO PREVIOUS FORSTK (5)
2219            *  11. OFFSET TO PREVIOUS GSBSTK (5)
2220            *  12. PREVIOUS PARAMETERS COUNT (2)
2221            *  13. OFFSET TO PREVIOUS PRMPTR (5)
2222            *  14. STSAVE (3)
2223            *  15. CHN#SV (2)
2224            *  16. RETURN TYPE (1)
2225            *                              TOTAL 105 NIBBLES
2226            *
2227 19C8C 78E1          GOSUB  GETPC
2228 19C90 168           DO=DO+ 9
2229 19C93 14E           C=DATO B
2230 19C96 96E           ?C#0   B              Is this a single line UDF ?
2231 19C99 11            GOYES  FNX495         If so, don't check CALC mode
2232            *
2233 19C9B 72D1          GOSUB  fcalc?         SEE IF IN CALC MODE
2234 19C9F 5A0           GONC   FNX495         IF NOT, OK
2235            *
2236 19CA2 3100          LC(2)  =eILCNT        SAY "ILLEGAL CONTEX"
2237 19CA6 6AFC  nferr2  GOTO   nferr
2238            *
2239 19CAA D2    FNX495  C=0    A
2240 19CAC 318C          LC(2)  200            NEED AT LEAST 200 NIBS TO DO FCN
2241 19CB0 74A1          GOSUB  A=D1           A= CURRENT TOP OF STACK
2242 19CB4 EA            A=A-C  A
2243 19CB6 1B49          DO=(5) =AVMEMS        CHECK IF WENT THROUGH AVMEMS
          5F2
2244 19CBD 146           C=DATO A
2245 19CC0 8B6           ?A>C   A
2246 19CC3 60            GOYES  FNX500
2247 19CC5 6FAD          GOTO   FNXNOR         IF SO,  SAY "INSUFFICIENT MEMORY"
2248 19CC9 1C0   FNX500  D1=D1- 1
2249 19CCC D2            C=0    A
2250 19CCE 87D           ?ST=1  13              RUNNING A PROGRAM ?
2251 19CD1 40            GOYES  FNX510
2252 19CD3 E6            C=C+1  A              TYPE = 1 - RETURN TO KEYBOARD
```

```
2253 19CD5 1550  FNX510 DAT1=C P
2254 19CD9 1BF6         DO=(5) =CHN#SV        SAVE CHN#SV(2)
           9F2
2255 19CE0 1C1          D1=D1- 2
2256 19CE3 14E          C=DATO B
2257 19CE6 14D          DAT1=C B
2258 19CE9 1BEB         DO=(5) =STSAVE        SAVE STSAVE
           6F2
2259 19CF0 146          C=DATO A
2260 19CF3 1C2          D1=D1- 3
2261 19CF6 1553         DAT1=C X
2262 19CFA 7E24         GOSUB  SAVPRM         SAVE PARAMETER COUNT & POINTER
2263 19CFE 1B3A         DO=(5) =GSBSTK        COMPUTE OFFSET OF PREVIOUS GSBSTK,
           5F2
2264 19D05 2D           P=     13
2265 19D07 7A34  FNX520 GOSUB  SAVPTR          FORSTK,MTHSTK AND SAVE THEM
2266 19D0B 184          DO=DO- 5              COMPUTE OFFSET OF PREVIOUS FORSTK
2267 19D0E 0C           P=P+1                  AND SAVE IT
2268 19D10 56F          GONC   FNX520
2269 19D13 1B18         DO=(5) =S-R1-0        SAVE S-R1-0,1,2,3
           8F2
2270 19D1A 1CF          D1=D1- 16
2271 19D1D 1567         C=DATO W
2272 19D21 1557         DAT1=C W
2273 19D25 18F          DO=DO- 16             SAVE S-R0-0,1,2,3
2274 19D28 1CF          D1=D1- 16
2275 19D2B 1567         C=DATO W
2276 19D2F 1557         DAT1=C W
2277 19D33 1B69         DO=(5) =STMTD1        SAVE STMTD1
           8F2
2278 19D3A 7C41         GOSUB  SAVE5-
2279 19D3E 2D           P=     13             SAVE 3 HARDEARE ARRESSES
2280 19D40 07    FNX530 C=RSTK
2281 19D42 7A41         GOSUB  SAVE5+
2282 19D46 0C           P=P+1
2283 19D48 57F          GONC   FNX530
2284 19D4B 184          DO=DO- 5              SAVE STMTDO
2285 19D4E 7B31         GOSUB  SAVE5
2286 19D52 7B2A         GOSUB  DOPCAD         SAVE PCADDR & CNTADR
2287 19D56 1C4          D1=D1- 5
2288 19D59 7C18         GOSUB  SAVE10
2289 19D5D 1B5A         DO=(5) =F-R0-2        POINTS TO SAVED PC
           8F2
2290 19D64 7221         GOSUB  SAVE5-
2291 19D68 D2           C=0    A              PUT 6 NIBS OF FOOOOO ON THE TOP
2292 19D6A 145          DAT1=C A
2293 19D6D 1C0          D1=D1- 1
2294 19D70 CE           C=C-1  A
2295 19D72 1550         DAT1=C P
2296 19D76 1B99         DO=(5) =MTHSTK
           5F2
2297 19D7D 133          AD1EX                 MOVE FORSTK & GSBSTK TO TOP OF STACK
2298               *
2299 19D80 2D           P=     13
2300 19D82 140   FNX535 DATO=A A
```

```
2301 19D85 164          D0=D0+ 5
2302 19D88 0C           P=P+1
2303 19D8A 57F          GONC   FNX535
2304              *
2305 19D8D 1BAA         D0=(5) =F-R0-3       UPDATE THE PRMPTR
           8F2
2306 19D94 D2           C=0    A
2307 19D96 1560         C=DAT0 P             C(B) = NEW PARAMETER COUNT
2308 19D9A E6           C=C+1  A
2309              *
2310              * IF  PARAMETER COUNT= 0   - NOT IN A DEF FN AT ALL
2311              *                    = 1   - IN A DEF FN BUT NO PARAMETER PASSED
2312              *                    = 1+N - IN A DEF FN WITH N PARAMETER PASSED
2313              *
2314 19D9C 1F7B         D1=(5) =PRMPTR
           5F2
2315 19DA3 14D          DAT1=C B
2316 19DA6 171          D1=D1+ 2
2317 19DA9 165          D0=D0+ 6             D0 POINTS TO F-R1-1
2318 19DAC 146          C=DAT0 A             C= SAVED NEW PRMPTR
2319 19DAF CB           C=C+D  A             REMEMBER WE EXTRACTED PARAM VALUES?
2320 19DB1 145          DAT1=C A             C= NEW PRMPTR
2321 19DB4 72B0         GOSUB  fntrck         NEED TO TRACE ?
2322 19DB8 4B1          GOC    FNX550         IF NOT, GOTO FNX550
2323 19DBB 8F00         GOSBVL =TRFROM
           000
2324 19DC2 72B0         GOSUB  GETPC
2325 19DC6 8F00         GOSBVL =COMPL#
           000
2326 19DCD 8F00         GOSBVL =TRTO*
           000
2327 19DD4 8E00  FNX550 GOSUBL =FPOLL
           00
2328 19DDA D3           CON(2) =pFNIN
2329 19DDC 7890         GOSUB  GETPC          UPDATE PCADDR
2330 19DE0 132          ADOEX
2331 19DE3 7A99         GOSUB  DOPCAD         D0=PCADDR
2332 19DE7 140          DAT0=A A
2333              * SET UP ADDRESS FOR TRACE
2334 19DEA D2           C=0    A
2335 19DEC 309          LC(1)  2+2+5
2336 19DEF CA           A=A+C  A
2337 19DF1 1FB8         D1=(5) =S-R1-2
           8F2
2338 19DF8 141          DAT1=A A
2339              *
2340 19DFB 1B5B         D0=(5) =F-R1-2       D0 POINTS TO F-R1-2
           8F2
2341 19E02 146          C=DAT0 A             C= SAVED PC
2342 19E05 134          D0=C
2343 19E08 108          R0=C
2344 19E0B 3100         LC(2)  =tCOMMA       SEE IF THIS IS A SINGLE LINE DEF FN
2345 19E0F 14A          A=DAT0 B
2346 19E12 966          ?A#C   B
2347 19E15 D3           GOYES  FNX590        IF NOT, GOTO FNX590
```

```
2348 19E17 119          C=R1                GET THE FUNCTION NAME FROM R1
2349 19E1A AB5          B=C    X
2350 19E1D 3102         LCHEX  20           SEE IF IT IS A STRING FUNCTION
2351 19E21 0E65         C=B&C  B
2352 19E25 96A          ?C=0   B            IS THIS A STRING FUNCTION ?
2353 19E28 11           GOYES  FNX560
2354 19E2A 743B         GOSUB  FCNADR
2355 19E2E 8F00         GOSBVL =PTRRCL      DO A STRING RECALL
           000
2356 19E35 6E00         GOTO   FNX570
2357 19E39 752B  FNX560 GOSUB  FCNADR
2358 19E3D 8F00         GOSBVL =RECALL
           000
2359               *
2360 19E44 85D   FNX570 ST=1   13
2361 19E47 8F00         GOSBVL =ASNSTO
           000
2362 19E4E 6B60         GOTO   FEND05       GOTO EXECUTE THE ENDFN
2363               *
2364               *
2365               *
2366               *
2367 19E52 8C7B  FNX590 GOLONG ENDS25       Set "PRGM" and clear "SUSP"
           7F
2368               *
2369 19E58 133   A=D1   AD1EX
2370 19E5B 131          D1=A
2371 19E5E 01           RTN
2372               *
2373 19E60 132   D0+C   AD0EX
2374 19E63 CA           A=A+C  A
2375 19E65 130          D0=A
2376 19E68 01           RTN
2377               *
2378 19E6A 737A  fntrck GOSUB  trflck
2379 19E6E 400          RTNC
2380 19E71 8D00  fcalc? GOVLNG =fCALC?
           000
2381               *
2382               *
2383               ****************************
2384 19E78 1BBA  GETPC  D0=(5)  =F-R1-0
           8F2
2385 19E7F 146          C=DAT0  A
2386 19E82 134          D0=C
2387 19E85 183          D0=D0-  4
2388 19E88 01           RTN
2389               *
2390 19E8A 1C4   SAVE5- D1=D1- 5
2391 19E8D 146   SAVE5  C=DAT0 A
2392 19E90 145   SAVE5+ DAT1=C A
2393 19E93 1C4          D1=D1- 5
2394 19E96 01           RTN
2395               *
2396 19E98 174   RST05+ D1=D1+ 5
```

```
2397 19E9B 147   RSTO5  C=DAT1 A
2398 19E9E 68D2         GOTO   RSTOP+
2399             ****************************************************************
2400             ****************************************************************
2401             **
2402             ** Name:    ENDDEF  -  END DEF statement execution
2403             **
2404             ** Category: STEXEC
2405             **
2406             ** Purpose: End an user-define function
2407             **
2408             ** Entry: Don't care
2409             **
2410             ** Exit: Exit to NXTSTM
2411             **
2412             ** Calls: FPOLL, GETCNT, TRFLCK, TRFROM, PRMCHN, DOPCAD
2413             **        RSTOPT, RSTPRM, MOVEMD, STMBUF, FCALC?, KBRTCK
2414             **        SNcrlf, SFLAGS, SFLAGC
2415             **
2416             ** Uses: A-D, R0-R3, D0,D1, P
2417             **
2418             ** Stk lvl: 4
2419             ****************************************************************
2420             ****************************************************************
2421             **
2422             *
2423 19EA2 0000         REL(5) =ENDDDC
          0
2424 19EA7               BSS    5
2425 19EAC 1F07 =ENDDEF D1=(5) =MLFFLG        SET MULTI-STMT USER FCN FLAG
          8F2
2426 19EB3 301           LC(1)  1
2427 19EB6 15D0          DAT1=C 1
2428             *
2429             *
2430 19EBA 8E00  FEND05 GOSUBL =FPOLL         Poll to see if anyone wants to
          00
2431 19EC0 E3            CON(2) =pFNOUT       do something.
2432             *
2433 19EC2 70B1          GOSUB  GETCNT
2434 19EC6 560           GONC   FEND10        IN A DEF FN
2435 19EC9 63DA          GOTO   FNMISS       IF SO, SAY "FN MISSING"
2436             *
2437 19ECD AE5   FEND10 B=C    B             B(B) = # OF PARAM OF THIS UDF
2438 19ED0 769F          GOSUB  fntrck       NEED TO TRACE ?
2439 19ED4 401           GOC    FEND15
2440 19ED7 8F00          GOSBVL =TRFROM
          000
2441 19EDE 7491          GOSUB  GETCNT
2442 19EE2 AE5           B=C    B
2443             *
2444             * NOW WE WANT TO LOOK FOR THE START OF THE UDF SAVE STACK.
2445             * STARTING FROM GSBSTK, GOING UP TO LOOK FOR F00000. IT SHOULD
2446             * BE AT THE BOTTOM OF GSBSTK. PAST THE GSBSTK IS THE START OF
2447             * THE UDF SAVE STACK.
```

```
2448                *
2449 19EE5 8F00     FEND15 GOSBVL =PRNCHN       LOOK FOR LAST OF GSBSTK
           000
2450 19EEC 147             C=DAT1 A             READ THE RETURN ADDRESS
2451 19EEF 10A             R2=C                 R2 = RETURN PC
2452 19EF2 7B88            GOSUB  DOPCAD        DO= PCADDR
2453 19EF6 174             D1=D1+ 5
2454 19EF9 8E29            GOSUBL RSTO10        Restore PCADDR & CNTADR
           6F
2455 19EFF 1B19            DO=(5) =STMTDO
           8F2
2456 19F06 719F            GOSUB  RSTO5          RESTORE STMTDO
2457 19F0A 2D              P=     13             RESTORE 3 HARDWARE ADDRESSES
2458 19F0C 147     FEND13  C=DAT1 A
2459 19F0F 06              RSTK=C
2460 19F11 174             D1=D1+ 5
2461 19F14 0C              P=P+1
2462 19F16 55F             GONC   FEND13
2463 19F19 164             DO=DO+ 5
2464 19F1C 7B7F            GOSUB  RSTO5
2465 19F20 1B17            DO=(5) =S-R0-0        RESTORE S-R0-0,1,2,3
           8F2
2466 19F27 1577            C=DAT1 W
2467 19F2B 1547            DAT0=C W
2468 19F2F 17F             D1=D1+ 16
2469 19F32 16F             DO=DO+ 16             RESTORE S-R1-0,1,2,3
2470 19F35 1577            C=DAT1 W
2471 19F39 1547            DAT0=C W
2472 19F3D 17F             D1=D1+ 16            POINTS TO SAVED FORSTK OFFSET
2473 19F40 1B99            DO=(5) =MTHSTK       RESTORE MTHSTK,FORSTK AND GSBSTK
           5F2
2474 19F47 2D              P=     13
2475 19F49 7122    FEND20  GOSUB  RSTOPT
2476 19F4D 164             DO=DO+ 5
2477 19F50 0C              P=P+1
2478 19F52 56F             GONC   FEND20
2479 19F55 7DF1            GOSUB  RSTPRM        RESTORE PARAMETER COUNT & POINTER
2480 19F59 1BEB            DO=(5) =STSAVE
           6F2
2481 19F60 147             C=DAT1 A
2482 19F63 1543            DAT0=C X             RESTORE STSAVE
2483 19F67 172             D1=D1+ 3
2484 19F6A 1BF6            DO=(5) =CHN#SV        RESTORE CHN#SV(2)
           9F2
2485 19F71 14F             C=DAT1 B
2486 19F74 14C             DAT0=C B
2487 19F77 171             D1=D1+ 2
2488 19F7A 11A             C=R2
2489 19F7D 1574            C=DAT1 S
2490 19F81 10A             R2=C                 R2(S) = RETURN TYPE
2491 19F84 DB              C=D    A             C= CURRENT PARAMETER CHAIN HEAD
2492 19F86 135             D1=C
2493 19F89 172     FEND30  D1=D1+ 3             PASS PARAMETER NAME
2494 19F8C A6D             B=B-1  B             PARAMETER CHAIN EXHAUSTED YET ?
2495 19F8F 480             GOC    FEND40        IF SO, GOTO FEND40
```

```
2496 19F92 17F          D1=D1+ 16
2497 19F95 53F          GONC   FEND30        (B.E.T.)
2498 19F98 30A  FEND40  LCHEX  A             SEE WHAT TYPE OF FUNCTION THIS IS
2499 19F9B 1537         A=DAT1 W
2500 19F9F 982          ?A<C   P             IS IT A REAL ?
2501 19FA2 97           GOYES  FEND60        IF SO, WE ARE DONE
2502 19FA4 AB8          B=A    X             MUST BE STRING OR COMPLEX FUNCTION
2503 19FA7 17A          D1=D1+ 11            POINTS TO ITS RELATIVE POINTER
2504 19FAA 147          C=DAT1 A             READ THE RELATIVE POINTER
2505 19FAD 77AE         GOSUB  A=D1
2506 19FB1 EE           C=A-C  A
2507 19FB3 134          D0=C                 D0 PTS TO STRING OR COMPLEX NUMBER
2508 19FB6 174          D1=D1+ 5             PASS THE DOPE VECTOR
2509 19FB9 D2           C=0    A
2510            * NOW D1 POINTS TO TOP OF EXPRESSION STACK
2511            *     D0 POINTS TO THE STRING OR COMPLEX NUMBER
2512 19FBB B05          B=B+1  P             TEST FOR STRING OR COMPLEX NUMBER
2513 19FBE 443          GOC    FEND50        IF IS STRING, GOTO FEND50
2514 19FC1 16F          D0=D0+ 16            POINTS TO END OF COMPLEX NUMBER
2515 19FC4 16F          D0=D0+ 16
2516 19FC7 3102         LC(2)  32            MOVE THE COMPLEX NUMBER
2517 19FCB 7D89         GOSUB  moved3
2518            *
2519            * Reverse the order of the imaginary and real of the complex
2520            * number.
2521            *
2522 19FCF 1577         C=DAT1 W
2523 19FD3 17F          D1=D1+ 16
2524 19FD6 1537         A=DAT1 W
2525 19FDA 1557         DAT1=C W
2526 19FDE 1CF          D1=D1- 16
2527 19FE1 1517         DAT1=A W
2528            *
2529 19FE5 1C1          D1=D1- 2
2530 19FE8 31E0         LCHEX  0E            CREATE COMPLEX NUMBER HEAD ON STACK
2531 19FEC 14D          DAT1=C B
2532 19FEF 6B20         GOTO   FEND60
2533 19FF3 15E3 FEND50  C=DAT0 4             READ STRING LENGTH
2534 19FF7 163          D0=D0+ 4             JUMP OVER STRING LENGTH
2535 19FFA C6           C=C+C  A             C= STRING LENGTH IN NIBBLES
2536 19FFC D5           B=C    A             B=     //
2537 19FFE 7E5E         GOSUB  D0+C          D0 POINTS TO END OF STRING
2538 1A002 7659         GOSUB  moved3        MOVE THE STRING
2539 1A006 1CF          D1=D1- 16            GENERATE STRING HEADER ON STACK
2540 1A009 AF2          C=0    W
2541 1A00C D9           C=B    A             C(A) = STRING LENGTH IN NIBBLE
2542 1A00E BF2          CSL    W
2543 1A011 BF2          CSL    W
2544 1A014 A0E          C=C-1  P
2545 1A017 1557         DAT1=C W
2546 1A01B 137  FEND60  CD1EX
2547 1A01E 10B          R3=C                 SAVE D1 IN R3
2548 1A021 8F00         GOSBVL =STMBUF       IF ENDDEF IS EXECUTED FROM KEYBOARD
           000
2549            *                            COLLAPSE THE STATEMENT BUFFER
```

```
2550 1A028 112          A=R2                A(S) = RETURN TYPE
2551 1A02B 948          ?A=0   S
2552 1A02E E1           GOYES  FEND80       RETURN TO PROGRAM FILE
2553              * RETURN TO KEYBOARD
2554 1A030 7D3E         GOSUB  fcalc?        CHECK CALC FLAG
2555 1A034 401          GOC    FEND75        CARRY SET IF RETURN TO CALC MODE
2556              *
2557              * KBRTCK will clear PgmRun, PRGM Annunciator and CNTADR
2558              *
2559 1A037 8F00         GOSBVL =SNcrlf
           000
2560 1A03E 8F00         GOSBVL =KBRTCK
           000
2561 1A045 84D   FEND75 ST=0   13
2562 1A048 6B10         GOTO   FEND90
2563              *
2564 1A04C 7A1E  FEND80 GOSUB  fntrck       NEED TO TRACE ?
2565 1A050 4F0          GOC    FEND85        IF NOT, GOTO FEND85
2566 1A053 11A          C=R2
2567 1A056 134          DO=C
2568 1A059 8F00         GOSBVL =TRTO-
           000
2569 1A060 7488  FEND85 GOSUB  SpgCsp       Set "PRGM" and clear "SUSP"
2570              *
2571 1A064 112   FEND90 A=R2
2572 1A067 130          DO=A
2573 1A06A 11B          C=R3
2574 1A06D 135          D1=C
2575 1A070 8C00 =expr   GOLONG =Expr
           00
2576              *
2577              ********************************************************************
2578              ********************************************************************
2579              **
2580              ** Name:    GETCNT  -  Get Function Parameter Counts
2581              **
2582              ** Category:   EXCUTL
2583              **
2584              ** Purpose:  Test whether we are in the middle of a user-defined
2585              **      function.
2586              **
2587              ** Entry:  Don't care
2588              **
2589              ** Exit:  C(B) = (PRMPTR)-1
2590              **
2591              ** Calls:  None
2592              **
2593              ** Uses:  C(B), D1
2594              **
2595              ** Stk lvls:  0
2596              ********************************************************************
2597              ********************************************************************
2598              *
2599 1A076 1F7B =GETCNT D1=(5) =PRMPTR
           5F2
```

```
2600 1A07D 14F          C=DAT1 B
2601 1A080 A6E          C=C-1  B
2602 1A083 01           RTN
2603            ****************************************************************
2604            ****************************************************************
2605            **
2606            ** Name:(S) GETNAM  -  Get variable name
2607            **
2608            ** Category:   VARMGT
2609            **
2610            ** Purpose:  Read the variable into B(X) and check if is a
2611            **           string or a number
2612            **
2613            ** Entry:  DO pts variable token
2614            **         P=0
2615            **
2616            ** Exit:  B(X) = Variable name
2617            **        DO past the variable name
2618            **        SO = 1 - is a string variable
2619            **             0 - is a numeric variable
2620            **        Carry set
2621            **
2622            ** Calls: ADRSUB
2623            **
2624            ** Uses:  B(A),C, SO, DO
2625            **
2626            ** Stk lvls:  +1
2627            ****************************************************************
2628            ****************************************************************
2629            **
2630 1A085 8F00 =GETNAM GOSBVL =ADRSUB       READ THE NAME
         000
2631 1A08C 840          ST=0   0            ASSUME IS A NUMERIC VARIABLE
2632 1A08F 3102         LCHEX  20
2633 1A093 0E65         C=C&B  B            TEST IF IS A STRING VARIABLE
2634 1A097 96A          ?C=0   B
2635 1A09A 00           RTNYES
2636 1A09C 850          ST=1   0            SET STRING FLAG
2637 1A09F 02           RTNSC
2638            *
2639            ****************************************************************
2640            ****************************************************************
2641            **
2642            ** Name:(S) FNDFCN  -  Find User-Defined Function
2643            **
2644            ** Category:   FILUTL
2645            **
2646            ** Purpose:  Find a user-defined function
2647            **
2648            ** Entry:  R1(X) = Function name(output from ADRSUB)
2649            **
2650            ** Exit:
2651            **      Carry set => Found
2652            **         DO past the function name in the DEF FN statement
2653            **         F-R1-O = Address past the tDEF of the DEF FN
```

```
2654                **      Carry clear => Not found
2655                **
2656                ** Calls:  PRSCOP, GETNAM
2657                **
2658                ** Uses:  A,B,C,D,D1,D0
2659                **
2660                ** Stk lvls:  4
2661                ********************************************************************
2662                ********************************************************************
2663                *
2664 1A0A1 8F00 =FNDFCN GOSBVL =PRSCKB       MAKE SURE PRGMST & PRGMEN CORRECT
           000
2665 1A0A8 1B26  FNDFN- D0=(5) =PRGMST
           5F2
2666 1A0AF 142          A=DAT0 A
2667 1A0B2 130          D0=A               D0 POINTS TO PRGMST
2668 1A0B5 184          D0=D0- 5           D0 @ TO 1ST LABEL LINK IN THIS PROG
2669 1A0B8 146          C=DAT0 A
2670 1A0BB D7           D=C    A
2671 1A0BD 1FBA         D1=(5) =F-R1-0
           8F2
2672 1A0C4 E7    FNX210 D=D+1  A           REACHED END OF CHAIN ?
2673 1A0C6 540          GONC   FNX220      NOT YET
2674 1A0C9 03           RTNCC              NOT FOUND
2675 1A0CB CF    FNX220 D=D-1  A
2676 1A0CD 136          CD0EX              GET TO NEXT LINK
2677 1A0D0 CB           C=C+D  A
2678 1A0D2 134          D0=C               D0 POINTS TO NEXT LINK
2679 1A0D5 146          C=DAT0 A           READ THE LABEL LINK
2680 1A0D8 D7           D=C    A           D= CURRENT LABEL LINK
2681 1A0DA 181          D0=D0- 2           D0 POINTS TO BEGIN TOKEN
2682 1A0DD 14A          A=DAT0 B
2683 1A0E0 161          D0=D0+ 2
2684 1A0E3 3100         LC(2)  =tDEF
2685 1A0E7 966          ?A#C   B           IS THIS A "DEF" TOKEN ?
2686 1A0EA AD           GOYES  FNX210      IF NOT, LOOK AT NEXT CHAIN
2687 1A0EC 132   FNX230 AD0EX              SAVE D0 IN A
2688 1A0EF 130          D0=A
2689 1A0F2 141          DAT1=A A
2690 1A0F5 166          D0=D0+ 7           D0 POINTS TO FUNCTION NAME
2691 1A0F8 798F         GOSUB  GETNAM
2692 1A0FC 119          C=R1
2693 1A0FF 931          ?B=C   X           IS THIS THE FUNCTION ?
2694 1A102 00           RTNYES             IF SO, RETURN WITH CARRY SET
2695 1A104 130          D0=A               RESTORE D0 FROM A
2696 1A107 5CB          GONC   FNX210      (B.E.T.)
2697                ***********************************
2698                *
2699                * CR-PAR - ROUTINE TO CREATE A PARAMETER
2700                *   INPUT:  B(X) = PARAMETER NAME
2701                *           D1 POINTS AT START OF PARAMETER NAME
2702                *           S0= 1 - STRING PARAMETER
2703                *               0 - NUMERIC PARAMETER
2704                *   EXIT:  D1 UNCHANGED
2705                *
```

```
2706 1A10A AB9   CR-PAR C=B    X
2707 1A10D 1553         DAT1=C X            WRITE THE PARAMETER NAME
2708 1A111 172          D1=D1+ 3
2709 1A114 AF2          C=0    W            CLEAR THE DOPE VECTOR
2710 1A117 860          ?ST=0  0            NUMERIC PARAMETER ?
2711 1A11A 90           GOYES  CR-P10       IF SO, GOTO CR-P10
2712 1A11C 34F0         LCHEX  2000F        C= 00000000002000F
          002
2713 1A123 1557  CR-P10 DAT1=C W            WRITE THE DOPE VECTOR
2714 1A127 1C2          D1=D1- 3            RESTORE D1
2715 1A12A 01           RTN
2716             *********************************
2717             * SAVPTR - SAVE THE CURRENT POINTER ON STACHK BY ITS OFFSET
2718             *
2719             *   INPUT:  D1= CURRENT STACK POINTER
2720             *           DO POINTS TO THE POINTER WILL BE SAVED
2721             *
2722 1A12C 1B9B  SAVPRM DO=(5) (=PRMPTR)+2  SAVE OFFSET OF "PRMPTR"
          5F2
2723 1A133 7E00         GOSUB  SAVPTR
2724 1A137 181          DO=DO- 2            SAVE PARAMETER COUNT
2725 1A13A 1C1          D1=D1- 2
2726 1A13D 14E          C=DAT0 B
2727 1A140 14D          DAT1=C B            SAVE PARAMETERS COUNT
2728 1A143 03           RTNCC
2729             *
2730 1A145 1C4   SAVPTR D1=D1- 5            RAISE THE STACK POINTER
2731 1A148 146          C=DAT0 A
2732 1A14B 790D         GOSUB  A=D1
2733 1A14F E2           C=C-A  A
2734 1A151 145          DAT1=C A
2735 1A154 03           RTNCC
2736             *************************************
2737             * RSTOPT - RESTORE POINTER. REVERSE THE PROCESS OF SAVPTR WHEN
2738             *            EXECUTE THE "ENDFN" STATEMENT
2739             *   INPUT:  DO POINTS TO THE POINTER BEING RESTORED
2740             *           D1 = CURRENT STACK POINTER
2741             *
2742 1A156 1B7B  RSTPRM DO=(5) =PRMPTR       DO POINTS TO PARAMETERS COUNT
          5F2
2743 1A15D 14F          C=DAT1 B            READ SAVED PARAMETRS COUNT
2744 1A160 14C          DAT0=C B            RESTORE PARAMETERS COUNT
2745 1A163 171          D1=D1+ 2            D1 POINTS TO SAVED PRMPTR
2746 1A166 161          DO=DO+ 2            DO POINTS TO SYSTEM PRMPTR
2747 1A169 146          C=DAT0 A            READ CURRENT PRMPTR AND SAVE IN D
2748 1A16C D7           D=C    A
2749             *
2750 1A16E 147  =RSTOPT C=DAT1 A            READ SAVED POINTEDR
2751 1A171 73EC         GOSUB  A=D1
2752 1A175 C2           C=A+C  A
2753 1A177 144   RSTOP+ DAT0=C A
2754 1A17A 174          D1=D1+ 5
2755 1A17D 01           RTN
2756             ***********************************
2757             * PTRSUB - ROUTINE TO GENERATE A DOPE VECTOR AND ALLOCATE SPACE
```

```
2758                *   INPUT:  C(A) = # OF NIBBLES TO BE ALLOCATED
2759                *           DO POINTS AT THE DOPE VECTOR
2760                *           R2= CURRENT STACK POINTER
2761                *           D1 POINTS TO PARAMETER VALUE ON THE STACK
2762                *   EXIT:   B(A) = POINTS TO NEXT PARAMETER
2763                *           R2= UPDATED STACK POINTER
2764                *           D1=       //
2765                *           DO POINTE TO NEXT PARAMETER VALUE ON THE STACK
2766                *
2767 1A17F 16A  PTRSUB DO=DO+ 11           POINTS TO RELATIVE POINTER
2768 1A182 DA          A=C    A           A= REQUIRED SPACE
2769 1A184 137         CD1EX              SAVE D1 IN R2
2770 1A187 12A         CR2EX              R2= SAVED D1, C= CURRENT STK.PTR
2771 1A18A E2          C=C-A  A
2772 1A18C 135         D1=C               D1= NEW STACK POINTER
2773 1A18F 8E00        GOSUBL =CHKSTK     SEE IF IS STILL ROOM
             00
2774 1A195 460         GOC    PTRS10      OK, IF CARRY SET
2775 1A198 6CD8        GOTO   FNXNOR      INSUFFICIENT MEMORY
2776 1A19C 136  PTRS10 CDOEX              A=NEW STK.PTR, C=REL.PTR ADDR
2777 1A19F 134         DO=C               DO= ADDR TO STORE REL.PTR
2778 1A1A2 E2          C=C-A  A           C= RELATIVE POINTER
2779 1A1A4 144         DAT0=C A           WRITE REL.PTR
2780 1A1A7 164         DO=DO+ 5           POINTS TO NEXT PARAMETER
2781 1A1AA 12A         CR2EX              C= SAVED D1
2782 1A1AD 102         R2=A               R2= NEW STACK POINTER
2783 1A1B0 136         CDOEX              DO POINTS TO PARAMETER VALUE
2784 1A1B3 D5          B=C    A           B(A) POINTS TO NEXT PARAMETER
2785 1A1B5 131         D1=A               D1= NEW STACK POINTER
2786 1A1B8 01          RTN
2787                *
2788                ********************************************************************
2789                ********************************************************************
2790                **
2791                ** Name:    SSB100  -  Search a Subprogram
2792                **
2793                ** Category:   FILUTL
2794                **
2795                ** Purpose: Search a Sub-Program
2796                **
2797                ** Entry:   R2 = SUBPROGRAM NAME
2798                **          (MTHSTK) = TOP OF CALL STACK
2799                ** Exit:
2800                **          CARRY CLEAR IF THE SUB-PROGRAM FOUND
2801                **            IF S3=0:  DO POINTS TO THE SUB-PROG NAME
2802                **            IF S3=1:  SUB-PROG IS A FILE, DO = PRGMST
2803                **          CARRY SET IF SUB-PROGRAM NOT FOUND
2804                **
2805                ** Calls: CHKTYP,SUBFIL, SCOPEN, PRSCOP, ROMCHK
2806                **        ROMFND
2807                **
2808                ** Uses: A-D, R0-R3, D1,D0, S2-S6,S8,S9
2809                **
2810                ** Stk lvls: +4
2811                **
```

```
2812                **
2813                ********************************************************************
2814                ********************************************************************
2815 1A1BA 843   SSB100 ST=0   3             FLAG SAYS SEARCHED FILE NAME
2816 1A1BD 845         ST=0   5             SEARECH ALL RAM/ROM
2817 1A1C0 76C1        GOSUB  CHKTP+        CURRENT FILE IS A BASIC/BINARY?
2818 1A1C4 11A         C=R2                 C = SUBPROGRAM NAME
2819 1A1C7 113         A=R3                 A = FILE HEADER ADDR
2820 1A1CA 5F5         GONC   SSB180        IF NOT, DON'T SEARCH CURRENT FILE
2821 1A1CD 97E         ?C#0   W             CALL WITH SUBPROGRAM NAME ?
2822 1A1D0 90          GOYES  SSB120        IF SO, GOTO SSB120
2823 1A1D2 853         ST=1   3
2824 1A1D5 63F0        GOTO   SSB280        CALL WITH NO SUBPROGRAM NAME
2825 1A1D9 7F11  SSB120 GOSUB  SUBFIL       SEARCH THE SUB-PROG IN CURRENT FILE
2826 1A1DD 495         GOC    SSB200        NOT FOUND IN CURRENT FILE
2827 1A1E0 136   SSB130 CDOEX
2828 1A1E3 134         D0=C
2829 1A1E6 135         D1=C
2830 1A1E9 1C4         D1=D1- 5             D1 POINTS TO SUB.LINK
2831 1A1EC 786C        GOSUB  A=D1          SET A=D1
2832 1A1F0 C0          A=A+B  A             A POINTS TO ENDSUB LINK
2833 1A1F2 1C3         D1=D1- 4             D1 POINTS TO LINE LENGTH
2834 1A1F5 D2          C=0    A
2835 1A1F7 14F         C=DAT1 B             C= LINE LENGTH
2836 1A1FA 133         AD1EX
2837 1A1FD C2          C=A+C  A             C POINTS TO END OF "SUB" STMT
2838 1A1FF 1C1         D1=D1- 2             D1 POINTS TO tSUB OR tENDSUB
2839                *
2840 1A202 E5          B=B+1  A             IS LINK # FFFFF ?
2841 1A204 590         GONC   SSB150
2842                *
2843                * Link = FFFFF, use end of file as end of program scope
2844                *
2845 1A207 DA          A=C    A             A= program start
2846 1A209 DB          C=D    A             C= END OF FILE
2847 1A20B 401         GOC    SSB160        (B.E.T)
2848                *
2849                * Link # FFFFF, D1 is at tSUB or tENDSUB
2850                *
2851 1A20E D5    SSB150 B=C    A             Save program start in B(A)
2852 1A210 8F00        GOSBVL =SCOPEN       Define the program scope end
           000
2853 1A217 D4          A=B    A             A= Program start
2854 1A219 137         CD1EX                C= Program end
2855                *
2856 1A21C 1F26  SSB160 D1=(5) =PRGMST
           5F2
2857 1A223 8D00        GOVLNG =PRSC60       SET PRGMST & PRGMEN
           000
2858                *
2859                * CURRENT FILE IS NOT A BASIC FILE
2860                *
2861 1A22A 97E   SSB180 ?C#0   W             CALL WITH NAME ?
2862 1A22D A0          GOYES  SSB200
2863 1A22F 3100        LC(2)  =eFTYPE
```

```
2864 1A233 627A          GOTO   mferr2
2865              *
2866              * SUB-PROG NOT FOUND IN CURRENT FILE
2867              *
2868 1A237 1F85  SSB200 D1=(5) =MAINST       SEARCH ENTIRE RAM FILES
           5F2
2869 1A23E 143           A=DAT1 A
2870 1A241 844           ST=0   4            FLAG TO SEARCH RAM
2871 1A244 7C41  SSB210 GOSUB  CHKTYP        IS FILE A BASIC/BINARY ?
2872 1A248 4A5           GOC    SSB260        IF SO, GOTO SSB260
2873              *
2874              * LOOK AT NEXT FILE
2875              *
2876 1A24B 875   SSB220 ?ST=1  5             FILE NAME SPECIFIED ?
2877 1A24E 00            RTNYES              IF SO, SAY NOT FOUND
2878 1A250 DB            C=D    A            C= FILE END ADDRESS
2879 1A252 DA            A=C    A
2880 1A254 135           D1=C
2881 1A257 14F           C=DAT1 B
2882 1A25A 96E           ?C#0   B            REACHED END OF RAM FILE ?
2883 1A25D 7E            GOYES  SSB210       IF NOT, GOTO SSB210
2884 1A25F 874           ?ST=1  4            ARE WE SEARCHING ROM ?
2885 1A262 D1            GOYES  SSB230       IF SO, KEEP SEARCHING
2886 1A264 854           ST=1   4            START SEARCHING ROM
2887 1A267 8F00          GOSBVL =ROMCHK
           000
2888 1A26E 4A1           GOC    SSB240
2889 1A271 133   SSB235 AD1EX
2890 1A274 303           LC(1)  3            SEE IF ROM OR IRAM
2891 1A277 816           CSRC
2892 1A27A 9C7           ?D<C   S
2893 1A27D 7C            GOYES  SSB210       IF SO, SEARCH IT
2894 1A27F 8F00  SSB230 GOSBVL =ROMFND
           000
2895 1A286 5AE           GONC   SSB235
2896              *
2897 1A289 873   SSB240 ?ST=1  3             HAVE WE LOOKED AT FILE NAME YET ?
2898 1A28C 00            RTNYES              IF SO, RETURN, SAY NOT FOUND
2899 1A28E 1FB4          D1=(5) =PRMCNT
           9F2
2900 1A295 14F           C=DAT1 B
2901 1A298 96E           ?C#0   B            IF THE CALL PASS ANY PARAM,
2902 1A29B 00            RTNYES                DON'T SEARCH IT AS A FILE NAME
2903 1A29D 853           ST=1   3            DENOTE WE WILL SEARCH IT AS A FILE
2904 1A2A0 569           GONC   SSB200       (B.E.T.)
2905              * FOUND A BASIC FILE
2906 1A2A3 113   SSB260 A=R3
2907 1A2A6 873           ?ST=1  3            SEARCHING FILE NAME ?
2908 1A2A9 11            GOYES  SSB270       YES
2909 1A2AB 7D40          GOSUB  SUBFIL       SEARCH THE SUB-PROG IN THIS FILE
2910 1A2AF 4B9   SSB265 GOC    SSB220        SUB-PROG NOT FOUND HERE
2911 1A2B2 7F20          GOSUB  SSB300       SET CURRST & CURREN
2912 1A2B6 692F          GOTO   SSB130       GOTO COMPUTE PRGMST & PRGMEN
2913              * SEARCH THE SUB-PROG AS A FILE
2914 1A2BA 131   SSB270 D1=A
```

```
2915 1A2BD 1577          C=DAT1 W
2916 1A2C1 112           A=R2
2917 1A2C4 976           ?A#C   W
2918 1A2C7 8E            GOYES  SSB265
2919 1A2C9 7810  SSB280  GOSUB  SSB300        SET CURRST & CURREN
2920 1A2CD 112           A=R2                 SAVE R2 IN R3
2921 1A2D0 103           R3=A
2922 1A2D3 8F00          GOSBVL =PRSCO-       COMPUTE PRGMST & PRGMEN
          000
2923 1A2DA 130           DO=A                 DO @ PRGMST
2924 1A2DD 11B           C=R3                 RESTORE R2 FROM R3
2925 1A2E0 10A           R2=C
2926 1A2E3 03            RTNCC
2927              *
2928 1A2E5 1FD5  SSB300  D1=(5) =CURRST       SET CURRST & CURREN IN RAM
          5F2
2929 1A2EC 113           A=R3
2930 1A2EF 141           DAT1=A A
2931 1A2F2 17E           D1=D1+ 15
2932 1A2F5 DB            C=D    A
2933 1A2F7 145           DAT1=C A
2934 1A2FA 01            RTN
2935              *
2936              ****************************************************************
2937              ****************************************************************
2938              **
2939              ** Name:    SUBFIL  -  Search a Subprogram Within a File
2940              **
2941              ** Category:   FILUTL
2942              **
2943              ** Purpose:  Search a subprogram within a BASIC or BINARY file
2944              **
2945              ** Entry:  A = Address of start of file header
2946              **         D = Address of end of file
2947              **         R2 = subprogram name
2948              **         R3 = Address of start of file header
2949              **         S8=1 - Binary file
2950              **         S8=0 - BASIC file
2951              **         P=0
2952              ** Exit:
2953              **         Carry set => Not found
2954              **         Carry clear => Found
2955              **          DO @ subprogram name
2956              **          B(A) = SUB.LINK
2957              **
2958              ** Calls:  LBLNAM, ISRAM?, CHAIN* (if the file is not chained)
2959              **
2960              ** Uses:  A, B, C, DO if the file is chained
2961              **        A-D, DO, D1, R0, R2, S9 if the file is not chained
2962              **
2963              ** Stk lvls:  1 if the file chained
2964              **            3 if the file is not chained
2965              ****************************************************************
2966              ****************************************************************
2967              *
```

```
2968 1A2FC 3452  SUBFIL LC(5)  =oSUBLn
           000
2969 1A303 CA            A=A+C  A
2970 1A305 130           D0=A                  D1 POINTS TO SUBLNK OF THE FILE
2971 1A308 146   SSB510  C=DAT0 A
2972 1A30B 8AE           ?C#0   A              IS THE FILE CHAINED YET ?
2973 1A30E 53            GOYES  SSB515         OF SO, GOTO SSB515
2974 1A310 878           ?ST=1  8               IS THIS A BINARY FILE ?
2975 1A313 00            RTNYES                 IF SO, RETURN, NOT FOUND
2976              * TEST IF THE FILE IS WRITEABLE
2977 1A315 D6            C=A    A
2978 1A317 8F00          GOSBVL =ISRAM?        CHECK IF THE FILE WRITEABLE
           000
2979 1A31E 440           GOC    SSB512
2980 1A321 02            RTNSC                 IF NOT RETURN WITH CARRY SET
2981 1A323 132   SSB512  ADOEX  A
2982 1A326 131           D1=A
2983 1A329 11A           C=R2                   SAVE R2 IN R0
2984 1A32C 108           R0=C
2985 1A32F 8F00          GOSBVL =CHAIN*
           000
2986 1A336 113           A=R3                   A= ADDRESS OF START OF FILE HEADER
2987 1A339 118           C=R0                   RESTORE R2 FROM R0
2988 1A33C 10A           R2=C
2989 1A33F 6CBF          GOTO   SUBFIL
2990              *
2991 1A343 E6    SSB515  C=C+1  A              SUBLNK= FFFFF ?
2992 1A345 400           RTNC                  IF SO, NO SUB-PROG EXIST
2993 1A348 CE            C=C-1  A
2994 1A34A 721B          GOSUB  D0+C
2995 1A34E 146   SSB520  C=DAT0 A
2996 1A351 D5            B=C    A              SAVE THE LINK IN B
2997 1A353 181           D0=D0- 2               SEE IF THIS IS A SUB STMT
2998 1A356 14A           A=DAT0 B
2999 1A359 3100          LC(2)  =tSUB
3000 1A35D 168           D0=D0+ 2+5+2          POINTS TO SUB NAME
3001 1A360 966           ?A#C   B
3002 1A363 11            GOYES  SSB525
3003 1A365 8F00          GOSBVL =LBLNAM
           000
3004 1A36C 11A           C=R2
3005 1A36F 972           ?A=C   W
3006 1A372 31            GOYES  SSB530
3007 1A374 E5    SSB525  B=B+1  A              REACHED END OF CHAIN ?
3008 1A376 400           RTNC                  NOT FOUND, IF SO
3009 1A379 D9            C=B    A
3010 1A37B 71EA          GOSUB  D0+C
3011 1A37F 187           D0=D0- 8              D0 POINTS TO NEXT LINK
3012 1A382 5BC           GONC   SSB520         (B.E.T.)
3013 1A385 181   SSB530  D0=D0- 2              D0 @ SUBPROGRAM NAME
3014 1A388 03            RTNCC
3015              *
3016              *
3017              ********************************************************************
3018              ********************************************************************
```

```
3019            **
3020            ** Name:    CHKTYP  -  Check file type
3021            **
3022            ** Category:   EXCUTL
3023            **
3024            ** Purpose:  Check if the file is a BASIC or BIN type
3025            **
3026            ** Entry:  A = Address of start of file header
3027            **
3028            ** Exit:  A(A) = File type
3029            **        D1 @ File type
3030            **        R3 = Address of start of file header
3031            **        D = Address of end of file
3032            **        Carry set => The file is a BASIC or BIN type
3033            **          S8 = 0 - BASIC
3034            **          S8 = 1 - BIN
3035            **        Carry clear => Not BASIC nor BIN
3036            **
3037            ** Calls : None
3038            **
3039            ** Uses:  A(A), C(A), D(A), R3, D1, S8
3040            **
3041            ** Stk lvls:  0
3042            ********************************************************************
3043            ********************************************************************
3044            *
3045 1A38A 1FD5  CHKTP+ D1=(5) =CURRST
         5F2
3046 1A391 143         A=DAT1 A
3047 1A394 103  =CHKTYP R3=A                  R3= CURRENT FILE START
3048 1A397 131         D1=A
3049 1A39A 17F         D1=D1+ (oFTYPh)
3050 1A39D 17F         D1=D1+ (oFLENh)-(oFTYPh)
3051 1A3A0 147         C=DAT1 A               C= FILE LENGTH
3052 1A3A3 71BA        GOSUB  A=D1
3053 1A3A7 C2          C=A+C  A
3054 1A3A9 D7          D=C    A               D= FILE END
3055 1A3AB 1CF         D1=D1- (oFLENh)-(oFTYPh)
3056 1A3AE D0          A=0    A
3057 1A3B0 15B3        A=DAT1 4
3058 1A3B4 858         ST=1   8
3059 1A3B7 3440        LC(5)  =fBIN
         2E0
3060 1A3BE 8A2         ?A=C   A               IS THIS A BASIC FILE ?
3061 1A3C1 00          RTNYES
3062 1A3C3 848         ST=0   8
3063 1A3C6 3441 =BASTYP LC(5)  =fBASIC
         2E0
3064 1A3CD 8A2         ?A=C   A
3065 1A3D0 00          RTNYES
3066 1A3D2 03          RTNCC
3067            *
3068            *
3069            *
3070 1A3D4             END
```

```
 A=D1    Abs  106072 #19E58 -  2369  1622  1653  2241  2505  2732  2751  2831
                               3052
 ACTIVE  Abs  193960 #2F5A8 -    13  1602
 ADJPTR  Ext                -  1186
 ADRS80  Ext                -  1827
 ADRSUB  Ext                -  2630
 ARYSIZ  Ext                -  1646
 ASNSTO  Ext                -  1957  2361
 AVMEMS  Abs  193940 #2F594 -    13  2243
=BASTYP  Abs  107462 #1A3C6 -  3063
 Binary  Abs      10 #0000A -   427   467   472   779
 BsErr   Ext                -  1509
 CAL010  Abs  101809 #18DB1 -   473   468
 CAL020  Abs  101827 #18DC3 -   483   476
 CAL030  Abs  101837 #18DCD -   485   488
 CAL040  Abs  101841 #18DD1 -   487   484
 CAL045  Abs  101847 #18DD7 -   489   481
 CAL050  Abs  101877 #18DF5 -   497   494
 CAL060  Abs  101900 #18E0C -   504   678
 CAL100  Abs  101956 #18E44 -   522   666
 CAL105  Abs  101971 #18E53 -   527   516   520
 CAL110  Abs  101975 #18E57 -   528   521   525
 CAL111  Abs  102047 #18E9F -   556   551
 CAL112  Abs  102053 #18EA5 -   559   540
 CAL114  Abs  102062 #18EAE -   614   555
 CAL115  Abs  102085 #18EC5 -   622   619
 CAL116  Abs  102089 #18EC9 -   623   621
 CAL120  Abs  102128 #18EF0 -   641   637
 CAL130  Abs  102154 #18F0A -   652   648
 CAL150  Abs  102178 #18F22 -   659   640   643   650
 CAL160  Abs  102188 #18F2C -   662
 CAL170  Abs  102193 #18F31 -   664   720
 CAL200  Abs  102212 #18F44 -   670   626
 CAL210  Abs  102239 #18F5F -   677   675
 CAL300  Abs  102249 #18F69 -   684   622
 CAL310  Abs  102288 #18F90 -   705   698
 CAL320  Abs  102317 #18FAD -   714   712
 CAL330  Abs  102320 #18FB0 -   715   692   706
 CAL335  Abs  102334 #18FBE -   720   718
 CAL400  Abs  102338 #18FC2 -   750   527
 CAL402  Abs  102375 #18FE7 -   763   759
 CAL404  Abs  102382 #18FEE -   765   756   762
 CAL405  Abs  102394 #18FFA -   768
 CAL410  Abs  102398 #18FFE -   769   767
 CAL412  Abs  102438 #19026 -   782   780
 CAL414  Abs  102445 #1902D -   786   783
 CAL415  Abs  102469 #19045 -   792   795
 CAL420  Abs  102543 #1908F -   813   810
 CAL422  Abs  102609 #190D1 -   831   828
 CAL424  Abs  102616 #190D8 -   833   821
 CAL425  Abs  102702 #1912E -   859   838
 CAL430  Abs  102711 #19137 -   862   855
 CAL450  Abs  102734 #1914E -   871   863   867
 CAL455  Abs  102759 #19167 -   877   880
 CAL460  Abs  102787 #19183 -   885   888
```

```
 CAL465  Abs  102816 #191A0 -   894   905
 CAL470  Abs  102853 #191C5 -   907   896
 CAL473  Abs  102875 #191DB -   916   924
 CAL475  Abs  102888 #191E8 -   920   914
 CAL478  Abs  102910 #191FE -   926   917
 CAL479  Abs  102925 #1920D -   931   929
 CAL480  Abs  102947 #19223 -   935  1098
 CAL485  Abs  102972 #1923C -   943   949
 CAL495  Abs  102976 #19240 -   944   942
 CAL500  Abs  102980 #19244 -   946   939
 CAL504  Abs  103006 #1925E -   954  1148
 CAL505  Abs  103010 #19262 -   955   918
 CAL510  Abs  103020 #1926C -   957   953
 CAL512  Abs  103076 #192A4 -   973   968
 CAL515  Abs  103085 #192AD -   976   960
 CAL518  Abs  103112 #192C8 -   984   975   982
 CAL519  Abs  103133 #192DD -  1012  1010
 CAL520  Abs  103142 #192E6 -  1017  1013
 CAL530  Abs  103177 #19309 -  1029  1024
 CAL550  Abs  103206 #19326 -  1038
 CAL560  Abs  103221 #19335 -  1043  1041
 CAL565  Abs  103224 #19338 -  1044  1067
 CAL570  Abs  103232 #19340 -  1046  1027
 CAL575  Abs  103240 #19348 -  1050  1061
 CAL580  Abs  103268 #19364 -  1059  1048
 CAL600  Abs  103300 #19384 -  1073  1014
 CAL620  Abs  103332 #193A4 -  1086
 CAL625  Abs  103366 #193C6 -  1095  1045  1058  1068  1109  1151
 CAL630  Abs  103377 #193D1 -  1099  1079
 CAL640  Abs  103407 #193EF -  1108  1131
 CAL650  Abs  103414 #193F6 -  1110  1107
 CAL660  Abs  103486 #1943E -  1135  1011
 CAL670  Abs  103526 #19466 -  1149  1147
 CAL710  Abs  103536 #19470 -  1160   943   983  1070  1136
 CAL800  Abs  103550 #1947E -  1168   944
 CAL810  Abs  103653 #194E5 -  1199  1204
 CAL830  Abs  103680 #19500 -  1209   930  1179
 CAL840  Abs  103732 #19534 -  1223  1220
=CALBIN  Abs  101772 #18D8C -   461
 CALERX  Abs  103296 #19380 -  1070  1030  1037  1051  1063  1074  1084  1100
 CALFX1  Ext                -   976
=CALL    Abs  101806 #18DAE -   472
 CALLDC  Ext                -   470
 CALLP   Ext                -   471
 CALNOR  Abs  103014 #19266 -   956   495   676   768
 CALSTK  Abs  193965 #2F5AD -    13   790  1187  1544  1584
 CHAIN*  Ext                -  2985
 CHKSPC  Ext                -   766   952
 CHKSTK  Ext                -   493  2773
 CHKTP+  Abs  107402 #1A38A -  3045  1218  2817
=CHKTYP  Abs  107412 #1A394 -  3047  2871
 CHN#SV  Abs  194927 #2F96F -    13  2254  2484
 CHNLST  Abs  193982 #2F5BE -    13   884  1591
 CLOSEA  Ext                -  1352
 CLPSTK  Ext                -  1351
```

```
 COMPL#  Ext                  -  2325
 CR-ARR  Ext                  -  1115
 CR-P10  Abs  106787 #1A123   -  2713  2711
 CR-PAR  Abs  106762 #1A10A   -  2706  1998  2031
 CR-VAR  Ext                  -  1017  1101
 CURRST  Abs  193885 #2F55D   -    13  1521  2928  3045
 CVALUE  Abs  105345 #19B81   -  2085  2071
 DO+C    Abs  106080 #19E60   -  2373  1753  2537  2994  3010
 DOPCAD  Abs  104321 #19781   -  1552   462  1524  2286  2331  2452
=D1FSTK  Abs  103773 #1955D   -  1281   491   507   813   848   962  1807
=D1MSTK  Abs  103758 #1954E   -  1257   543   633   670   689
 D1PRCT  Abs  101763 #18D83   -   303   504   614   889   935
 D1SUB   Abs  103765 #19555   -  1258  1282
=DEF     Abs  104697 #198F9   -  1744
 DEF10   Abs  104713 #19909   -  1750  1762  1767  1769  1771
 DEF30   Abs  104772 #19944   -  1770  1759
 DEF60   Abs  104788 #19954   -  1775  1747
 DEF70   Abs  104792 #19958   -  1777  1752
 DEFDC   Ext                  -  1742
 DEFP    Ext                  -  1743
 DEST    Ext                  -   667
 DFLTCR  Ext                  -  1086
 END     Ext                  -   301
 END10   Ext                  -  1325
 ENDDDC  Ext                  -  2423
=ENDDEF  Abs  106156 #19EAC   -  2425
 ENDS20  Abs  103906 #195E2   -  1334  1328
 ENDS25  Abs  103947 #1960B   -  1346  2367
 ENDS30  Abs  103955 #19613   -  1351  1335  1344
 ENDS40  Abs  103976 #19628   -  1358  1342
 ENDS45  Abs  103983 #1962F   -  1360  1347
 ENDS46  Abs  104005 #19645   -  1371  1374
 ENDS50  Abs  104026 #1965A   -  1379  1221  1340
=ENDSB-  Abs  103855 #195AF   -  1326
 ENDSDC  Ext                  -  1322
=ENDSUB  Abs  103848 #195A8   -  1325  1777
 EOLXCK  Ext                  -   515
 ERRSUB  Abs  194179 #2F683   -    13  1213
 EXPCH#  Ext                  -   916  1146
 Expr    Ext                  -  2575
 F#NFND  Ext                  -   556
 F-RO-1  Abs  194720 #2F8A0   -    13  1990
 F-RO-2  Abs  194725 #2F8A5   -    13  2289
 F-RO-3  Abs  194730 #2F8AA   -    13  1964  2305
 F-R1-0  Abs  194731 #2F8AB   -    13  2384  2671
 F-R1-2  Abs  194741 #2F8B5   -    13  2037  2340
 F-R1-3  Abs  194746 #2F8BA   -    13  1034
 FCNADR  Abs  104802 #19962   -  1807  1949  2354  2357
 FDCH#   Ext                  -   550
 FEND05  Abs  106170 #19EBA   -  2430  2362
 FEND10  Abs  106189 #19ECD   -  2437  2434
 FEND13  Abs  106252 #19F0C   -  2458  2462
 FEND15  Abs  106213 #19EE5   -  2449  2439
 FEND20  Abs  106313 #19F49   -  2475  2478
 FEND30  Abs  106377 #19F89   -  2493  2497
```

```
 FEND40  Abs  106392 #19F98 -  2498  2495
 FEND50  Abs  106483 #19FF3 -  2533  2513
 FEND60  Abs  106523 #1A01B -  2546  2501  2532
 FEND75  Abs  106565 #1A045 -  2561  2555
 FEND80  Abs  106572 #1A04C -  2564  2552
 FEND85  Abs  106592 #1A060 -  2569  2565
 FEND90  Abs  106596 #1A064 -  2571  2562
 FILXQ^  Ext                -   483
 FINDF+  Ext                -   845
 FLTDH   Ext                -  2009
=FN      Abs  104894 #199BE -  1934
=FNDFCN  Abs  106657 #1A0A1 -  2664
 FNDFN-  Abs  106664 #1A0A8 -  2665  1987
 FNDMK-  Ext                -   920
=FNMISS  Abs  104861 #1999D -  1825  1812  1989  2435
 FNX110  Abs  104837 #19985 -  1817  1821
 FNX120  Abs  104852 #19994 -  1822  1818
 FNX130  Abs  104871 #199A7 -  1827  1824
 FNX140  Abs  104953 #199F9 -  1955  1952
 FNX150  Abs  104960 #19A00 -  1957  1954
 FNX200  Abs  104973 #19A0D -  1961  1940
 FNX206  Abs  105037 #19A4D -  1986  1979
 FNX210  Abs  106692 #1A0C4 -  2672  2686  2696
 FNX220  Abs  106699 #1A0CB -  2675  2673
 FNX230  Abs  106732 #1A0EC -  2687
 FNX240  Abs  105055 #19A5F -  1990  1988
 FNX260  Abs  105083 #19A7B -  1996  1994
 FNX265  Abs  105134 #19AAE -  2012  2018
 FNX270  Abs  105140 #19AB4 -  2014  2010
 FNX275  Abs  105147 #19ABB -  2017  2016
 FNX280  Abs  105162 #19ACA -  2022  2000  2004
 FNX300  Abs  105180 #19ADC -  2028  2035
 FNX310  Abs  105212 #19AFC -  2037  2026
 FNX320  Abs  105252 #19B24 -  2055  2084  2133
 FNX330  Abs  105266 #19B32 -  2059  2057
 FNX380  Abs  105452 #19BEC -  2129  2105
 FNX400  Abs  105470 #19BFE -  2138  2058
 FNX410  Abs  105506 #19C22 -  2150  2147
 FNX420  Abs  105521 #19C31 -  2154  2145
 FNX430  Abs  105539 #19C43 -  2161  2153
 FNX435  Abs  105566 #19C5E -  2170  2165
 FNX450  Abs  105574 #19C66 -  2177  2149
 FNX490  Abs  105587 #19C73 -  2182  2179
 FNX495  Abs  105642 #19CAA -  2239  2231  2234
 FNX500  Abs  105673 #19CC9 -  2248  2246
 FNX510  Abs  105685 #19CD5 -  2253  2251
 FNX520  Abs  105735 #19D07 -  2265  2268
 FNX530  Abs  105792 #19D40 -  2280  2283
 FNX535  Abs  105858 #19D82 -  2300  2303
 FNX550  Abs  105940 #19DD4 -  2327  2322
 FNX560  Abs  106041 #19E39 -  2357  2353
 FNX570  Abs  106052 #19E44 -  2360  2356
 FNX590  Abs  106066 #19E52 -  2367  2347
 FNXNOR  Abs  105077 #19A75 -  1995   956  2030  2247  2775
 FNXTER  Abs  105318 #19B66 -  2077  2062  2080  2088  2108  2180
```

```
 FORSTK  Abs  193950 #2F59E -    13  1170  1281  1530
 FPOLL   Ext                -  2327  2430
 GETC#-  Ext                -   542
=GETCNT  Abs  106614 #1A076 -  2599  2433  2441
=GETNAM  Abs  106629 #1A085 -  2630  1936  2028  2691
 GETPC   Abs  106104 #19E78 -  2384  2227  2324  2329
 GETST   Ext                -   490
 GETSTC  Ext                -   758
 GSBSTK  Abs  193955 #2F5A3 -    13  2263
 I/OALL  Ext                -   901
 I/OFND  Ext                -   895
 ISRAM?  Ext                -  2978
=InvArg  Abs  103070 #1929E -   971  2012
 Invarg  Ext                -   971
 KBRTCK  Ext                -  1358  2560
 LBLNAM  Ext                -  3003
 LNSKP-  Ext                -  1336
 MAINST  Abs  193880 #2F558 -    13  2868
=MANSTK  Abs  104147 #196D3 -  1472  1192  1482  1513
 MEMERj  Ext                -  1995
 MFERRj  Ext                -  1826
 MISUBP  Abs  102694 #19126 -   856   485   860
 MLFFLG  Abs  194672 #2F870 -    13  2425
 MOVED3  Ext                -  1780
 MOVEU3  Ext                -  2129
=MSTKD1  Abs  103740 #1953C -  1251   665   806
 MTHSTK  Abs  193945 #2F599 -    13   770   873  1252  1257  2296  2473
 NEWVAR  Ext                -   673
=NOTMCH  Abs  103542 #19476 -  1162  2077
 NXPM10  Abs  104064 #19680 -  1420  1427  1437  1443
 NXPM15  Abs  104081 #19691 -  1426
 NXPM20  Abs  104087 #19697 -  1428  1425
 NXPM50  Abs  104134 #196C6 -  1444  1423
 NXTPRM  Abs  104036 #19664 -  1413   986
 NXTST2  Ext                -  1774
 NXtstm  Ext                -  1959
 ONINTR  Abs  194189 #2F68D -    13   796
 PCADDR  Abs  194169 #2F679 -    13  1552
 POLL    Ext                -   807  1505
 POPARG  Ext                -   964
 POPCH#  Ext                -  1550
 POPST-  Abs  104334 #1978E -  1581   955   970
 POPSTK  Abs  104330 #1978A -  1580  1161  1334
 POPSTR  Ext                -   657
 PRGMEN  Abs  193895 #2F567 -    13   800
 PRGMST  Abs  193890 #2F562 -    13  1209  2665  2856
 PRMCHN  Ext                -  2449
 PRMCNT  Abs  194891 #2F94B -    13   303  1413  2899
 PRMPTR  Abs  193975 #2F5B7 -    13  1809  2314  2599  2722  2742
 PRSC00  Ext                -   763
 PRSC60  Ext                -  2857
 PRSCKB  Ext                -  1986  2664
 PRSC0-  Ext                -  2922
 PTRRCL  Ext                -  1953  2355
 PTRS10  Abs  106908 #1A19C -  2776  2774
```

```
 PTRSUB  Abs  106879 #1A17F -  2767  2091  2117  2161
 RAMEND  Abs  193970 #2F5B2 -    13  1542  1544
 RBC100  Abs  104379 #197BB -  1594  1597
 RBC110  Abs  104402 #197D2 -  1601  1664
 RBC115  Abs  104427 #197EB -  1608  1606
 RBC125  Abs  104447 #197FF -  1619  1612  1700
 RBC130  Abs  104464 #19810 -  1625  1620
 RBC140  Abs  104554 #1986A -  1660  1658
 RBC150  Abs  104560 #19870 -  1662  1637  1640  1643
 RBC155  Abs  104563 #19873 -  1663  1678
 RBC160  Abs  104566 #19876 -  1664
 RBC200  Abs  104570 #1987A -  1668  1613  1624
 RBC210  Abs  104601 #19899 -  1679  1669
 RBC220  Abs  104612 #198A4 -  1682  1699  1703
 RBC230  Abs  104629 #198B5 -  1688  1686
 RBC240  Abs  104668 #198DC -  1702  1607
=RBLDCH  Abs  104341 #19795 -  1584
 RDATTY  Ext                -   703
 RECALL  Ext                -  1955  2358
 RESTDO  Ext                -   677
 ROMCHK  Ext                -  2887
 ROMFND  Ext                -  2894
 RST010  Abs  103825 #19591 -  1299  1525  2454
 RST020  Abs  103811 #19583 -  1295  1522  1528
 RST05   Abs  106139 #19E9B -  2397  2456  2464
 RST05+  Abs  106136 #19E98 -  2396
 RSTOP+  Abs  106871 #1A177 -  2753  2398
=RSTOPT  Abs  106862 #1A16E -  2750  1532  2475
 RSTP10  Abs  104187 #196FB -  1505
 RSTP20  Abs  104204 #1970C -  1511  1507
 RSTP50  Abs  104264 #19748 -  1532  1535
 RSTPRM  Abs  106838 #1A156 -  2742  1536  2479
 RSTPTR  Abs  104180 #196F4 -  1502  1581
 RUNRT1  Ext                -  1377
 RUSUS?  Ext                -   755
 RVALUE  Abs  105322 #19B6A -  2078  2067
 S-R0-0  Abs  194673 #2F871 -    13   771   931  2465
 S-R1-0  Abs  194689 #2F881 -    13  2269
 S-R1-2  Abs  194699 #2F88B -    13  1944  2337
 SAV10+  Abs  103795 #19573 -  1289   799
 SAVE10  Abs  103801 #19579 -  1291   772  2288
 SAVE20  Abs  103784 #19568 -  1286   797   801
 SAVE5   Abs  106125 #19E8D -  2391  2285
 SAVE5+  Abs  106128 #19E90 -  2392  2281
 SAVE5-  Abs  106122 #19E8A -  2390  2278  2290
 SAVEDO  Ext                -   497
 SAVPRM  Abs  106796 #1A12C -  2722   787  2262
 SAVPTR  Abs  106821 #1A145 -  2730   792  2265  2723
 SCOPCK  Ext                -  1343
 SCOPEN  Ext                -  2852
 SFGPGM  Ext                -  1710
 SFLAG?  Ext                -  1978
 SFLAGS  Ext                -   502
 SFlagC  Ext                -   872
 SKITM2  Abs  104116 #196B4 -  1438  1430
```

```
 SKITM5  Abs  104128 #196C0 -  1442  1439
 SKNM10  Abs  102720 #19140 -   865   869
 SNcrlf  Ext                -  2559
 SSB100  Abs  106938 #1A1BA -  2815   859
 SSB120  Abs  106969 #1A1D9 -  2825  2822
 SSB130  Abs  106976 #1A1E0 -  2827  2912
 SSB150  Abs  107022 #1A20E -  2851  2841
 SSB160  Abs  107036 #1A21C -  2856  2847
 SSB180  Abs  107050 #1A22A -  2861  2820
 SSB20   Abs  102650 #190FA -   845   841
 SSB200  Abs  107063 #1A237 -  2868  2826  2862  2904
 SSB210  Abs  107076 #1A244 -  2871   854  2883  2893
 SSB220  Abs  107083 #1A24B -  2876  2910
 SSB230  Abs  107135 #1A27F -  2894  2885
 SSB235  Abs  107121 #1A271 -  2889  2895
 SSB240  Abs  107145 #1A289 -  2897  2888
 SSB260  Abs  107171 #1A2A3 -  2906  2872
 SSB265  Abs  107183 #1A2AF -  2910  2918
 SSB270  Abs  107194 #1A2BA -  2914  2908
 SSB280  Abs  107209 #1A2C9 -  2919  2824
 SSB300  Abs  107237 #1A2E5 -  2928  2911  2919
 SSB510  Abs  107272 #1A308 -  2971
 SSB512  Abs  107299 #1A323 -  2981  2979
 SSB515  Abs  107331 #1A343 -  2991  2973
 SSB520  Abs  107342 #1A34E -  2995  3012
 SSB525  Abs  107380 #1A374 -  3007  3002
 SSB530  Abs  107397 #1A385 -  3013  3006
 SSBER   Abs  102646 #190F6 -   842   846
 STK19?  Ext                -  1993  2029
 STMBUF  Ext                -  1326  2548
 STMTD0  Abs  194705 #2F891 -    13   933  1168  1416  2455
 STMTD1  Abs  194710 #2F896 -    13  2277
 STPASG  Ext                -   900
 STRASN  Ext                -  1094
 STROVF  Ext                -   713
 STSAVE  Abs  194238 #2F6BE -    13  2258  2480
=SUB     Abs  101756 #18D7C -   301
=SUBCHK  Abs  104292 #19764 -  1542  1502  1511
 SUBDC   Ext                -   299
 SUBFIL  Abs  107260 #1A2FC -  2968  2825  2909  2989
 SUBP    Ext                -   300
 SVALUE  Abs  105399 #19BB7 -  2106  2076
 SpgCsp  Abs  104680 #198E8 -  1710  1223  1346  2569
 TRCLIN  Ext                -   822  1329
 TRFLCK  Ext                -  1705
 TRFROM  Ext                -  2323  2440
 TRTO*   Ext                -  2326
 TRTO-   Ext                -  2568
 TRTOEN  Ext                -   831  1333
 bASSGN  Abs    2052 #00804 -    12   894
 bserr   Abs  104198 #19706 -  1509   811   842
=dest    Abs  102205 #18F3D -   667   671   977
 eFNNtF  Ext                -  1825
 eFTYPE  Ext                -  2863
 eFnFND  Ext                -  1353
```

```
 eILCNT  Ext                  -  2236
 eILPAR  Ext                  -  1981
 ePRMIS  Ext                  -  1162
 eSPGNF  Ext                  -   856
=expr    Abs  106608 #1A070  -  2575   559   963  2006
 fBASIC  Abs   57876 #0E214  -    12  3063
 fBIN    Abs   57860 #0E204  -    12   760  3059
 fCALC?  Ext                  -  2380
 fcalc?  Abs  106097 #19E71  -  2380  2233  2554
 flNOFN  Abs     -42 #FFFD6  -    12   501   871  1977
 fntrck  Abs  106090 #19E6A  -  2378  2321  2438  2564
 fspecx  Ext                  -   840
 mferl   Abs  103546 #1947A  -  1163   857  1354
 mferr   Abs  104865 #199A1  -  1826  1163  1982  2237
 mferr2  Abs  105638 #19CA6  -  2237  2864
 moved3  Abs  104796 #1995C  -  1780  1183  2190  2517  2538
 noroom  Abs  102884 #191E4  -   918   902
 nxtstm  Abs  104967 #19A07  -  1959  1775
 oFLENh  Abs      32 #00020  -    12  3050  3055
 oFTYPh  Abs      16 #00010  -    12  3049  3050  3055
 oSUBLn  Abs      37 #00025  -    12  2968
 pCALRS  Abs      54 #00036  -    12  1506
 pCALSV  Abs      55 #00037  -    12   808
 pFNIN   Abs      61 #0003D  -    12  2328
 pFNOUT  Abs      62 #0003E  -    12  2431
 popch#  Abs  104315 #1977B  -  1550   954   969  1580
 rstst   Ext                  -   769
 runrtl  Abs  104019 #19653  -  1377  1366
 runrtn  Abs  103989 #19635  -  1363  1224
 t!      Ext                  -  1363
 tCOMMA  Ext                  -  2344
 tDEF    Ext                  -  1760  2684
 tENDDF  Ext                  -  1755
 tEOL    Ext                  -   474
 tIN     Ext                  -   836
 tPRMEN  Ext                  -   523   937
 tPRMST  Ext                  -   518  2023
 tSEMIC  Ext                  -  2001
 tSUB    Ext                  -  2999
 trflck  Abs  104673 #198E1  -  1705   820  1327  2378
```

Input Parameters

  Source file name is SC&SUB::MS

  Listing file name is SC/SUB:TI:ML::-1

  Object file name is SC%SUB:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                *     SSS   BBBB    &     EEEEE  X   X   CCC
 2                *    S   S  B   B  & &    E      X   X  C   C
 3                *    S      B   B  & &    E       X X   C
 4                *     SSS   BBBB    &     EEEE     X    C
 5                *        S  B   B  & & &  E       X X   C
 6                *    S   S  B   B  &  &   E      X   X  C   C
 7                *     SSS   BBBB    && &  EEEEE  X   X   CCC
 8                *
 9                       TITLE  Miscellaneous Execution Routines <831212.1206>
10 1A3D4         SBEXC  ABS    #1A3D4
11                       RDSYMB TIXEQU::MS
12                *-
13                *-
14                ********************************************************************
15                ********************************************************************
16                **
17                ** Name:    ENDLIN  -  ENDLINE statement execution
18                **
19                ** Category:   STEXEC
20                **
21                ** Purpose:
22                **      Implements ENDLINE statement
23                **
24                ** Entry:
25                **      P      =  0
26                **      DO points past ENDLINE token
27                **
28                ** Exit:
29                **      Exits through NXTSTM
30                **
31                ** Calls:      EXPEXC,POP1S,NXTSTM
32                **
33                ** Algorithm:
34                **      If no expression present then
35                **        set endline string to CR/LF
36                **      else
37                **        Evaluate expression
38                **        Pop string from stack
39                **        If length>3 bytes then
40                **          exit with "Invalid Argument" error
41                **        Set endline string and string length
42                **      Exit through NXTSTM
43                **
44                ** History:
45                **
46                **    Date      Programmer              Modification
47                **  --------   ----------   ----------------------------------
48                **  10/25/83   B.S.         Added documentation
49                **
50                ********************************************************************
51                ********************************************************************
52 1A3D4 8C00     invarg GOLONG =InvArg
          00
53                *-
54                *-
```

```
     55 1A3DA 0000          REL(5) =DROPDC
              0
     56 1A3DF 0000          REL(5) =OPSTRp
              0
     57 1A3E4 14A  =ENDLIN  A=DAT0 B
     58 1A3E7 310F          LCHEX  F0
     59 1A3EB 9E2           ?A<C   B              Is there a parameter?
     60 1A3EE 61            GOYES  ENDL00         Yes, then evaluate it
     61 1A3F0 1F00          D1=(5) =EOLLEN        No, then set to default (CR/LF)
              000
     62 1A3F7 344D          LCHEX  0A0D4
              0A0
     63 1A3FE 145           DAT1=C A
     64 1A401 584           GONC   nxtstm         (B.E.T.)
     65               *-
     66               *-
     67 1A404 8E00  ENDL00  GOSUBL =expexc        Evaluate expression
              00
     68 1A40A 8F00          GOSBVL =POP1S         Pop string off stack
              000
     69 1A411 AF2           C=0    W
     70 1A414 306           LC(1)  6
     71 1A417 8B6           ?A>C   A              Is string > 3 chars?
     72 1A41A AB            GOYES  invarg         Yes, then error (invalid arg)
     73 1A41C 1B00          D0=(5) =EOLLEN
              000
     74 1A423 1500          DAT0=A P              Write out EOL len
     75 1A427 160           D0=D0+ 1
     76 1A42A 137           CD1EX
     77 1A42D C2            C=C+A  A
     78 1A42F 135           D1=C
     79 1A432 81C           ASRB                  Divide by 2 for number of chars
     80 1A435 A0C   ENDL10  A=A-1  P              Decrement count
     81 1A438 411           GOC    ENDL20         Exit loop if done
     82 1A43B 1C1           D1=D1- 2
     83 1A43E 14F           C=DAT1 B              Read character from string
     84 1A441 14C           DAT0=C B              Copy it to EOLSTR
     85 1A444 161           D0=D0+ 2              Move EOLSTR pointer
     86 1A447 5DE           GONC   ENDL10         (B.E.T.) Loop until done
     87 1A44A       ENDL20
     88 1A44A 6000  nxtstm  GOTO   =REN180        GOTO NXTSTM
     89               *-
     90               *-
```

```
 91                     EJECT
 92                 ****************************************************************
 93                 ****************************************************************
 94                 **
 95                 ** Name:(S) XXHEAD  -  Remove String Header (Undo ADHEAD)
 96                 **
 97                 ** Category:   MTHSTK
 98                 **
 99                 ** Purpose:
100                 **      Removes string header from a string on stack.  Leaves
101                 **      registers set up so that STKCHR may be called again.
102                 **
103                 ** Entry:
104                 **      P      =  0
105                 **
106                 ** Exit:
107                 **      P      =  0
108                 **      D(A)=Pointer to AVMEMS
109                 **      R1(A)=Pointer to end of stack item (highest address)
110                 **      D1 points to start of stack item (lowest address)
111                 **      Carry clear
112                 **
113                 ** Calls:      POP1S
114                 **
115                 ** Uses.......
116                 **  Inclusive: C(A),D1,D(A)
117                 **
118                 ** Stk lvls:   1
119                 **
120                 ** History:
121                 **
122                 **    Date      Programmer              Modification
123                 **  --------   ----------   ----------------------------------
124                 **  10/19/82   B.S.         Added documentation
125                 **
126                 ****************************************************************
127                 ****************************************************************
128 1A44E 8F00 =XXHEAD GOSBVL =POP1S
          000
129 1A455 137           CD1EX
130 1A458 135           D1=C
131 1A45B C2            C=C+A  A
132 1A45D 109           R1=C
133                 *Now fall into D=AVMS
```

```
                        EJECT
                ****************************************************************
                ****************************************************************
                **
                ** Name:(S) D=AVMS  -  Set D(A) to AVMEMS or AVMEME
                ** Name:(S) D=AVME  -  Set D(A) to AVMEMS or AVMEME
                **
                ** Category:   PTRUTL
                **
                ** Purpose:
                **      D=AVMS  :  Read AVMEMS into D(A)
                **      D=AVME  :  Read AVMEME into D(A)
                **
                ** Entry:
                **
                ** Exit:
                **      D(A)=memory location specified.
                **      C(A)=a copy of value in D1 at time of call
                **
                ** Calls:      None
                **
                ** Uses.......
                **  Inclusive: C(A),D(A)
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **   --------   ----------   ----------------------------------
                **   10/19/82   B.S.         Added documentation
                **
                ****************************************************************
                ****************************************************************
                *NOTE:  Above code falls through to here
1A460 137       =D=AVMS CD1EX
1A463 1F00              D1=(5) =AVMEMS
      000
1A46A D7         D=@D1  D=C    A
1A46C 147               C=DAT1 A
1A46F DF                CDEX   A
1A471 135               D1=C
1A474 01                RTN                    Preserve carry
                *-
                *-
1A476 137       =D=AVME CD1EX
1A479 1F00              D1=(5) =AVMEME
      000
1A480 69EF              GOTO   D=@D1
                *-
                *-
```

```
                        EJECT
                *****************************************************************
                *****************************************************************
                **
                ** Name:    STRTUP  -  STARTUP statement execution
                **
                ** Category:   STEXEC
                **
                ** Purpose:
                **      STARTUP statement execution
                **
                ** Entry:
                **      P      =  0
                **      DO points past STARTUP token
                **
                ** Exit:
                **      Exits through NXTSTM
                **
                ** Calls:      EXPEXC,XXHEAD,STROVF,ADHEAD
                **
                ** Algorithm:
                **      Evaluate expression
                **      Remove string header
                **      Append CR to string
                **      If string > 95 chars then
                **        Exit with "String Overflow" error
                **      If string is null then
                **        Deallocate buffer (bSTART)
                **        Exit through NXTSTM
                **      Allocate buffer of required length
                **      If not enough memory to hold buffer then
                **        Exit with "Insufficient Memory" error
                **      Copy startup string into buffer
                **      Exit through NXTSTM
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------   ---------------------------------
                **  10/25/83   B.S.         Added documentation
                **
                *****************************************************************
                *****************************************************************
1A484 0000      0       REL(5) =DELAYd
1A489 0000      0       REL(5) =STRNGP
1A48E           =STRTUP
1A48E 8E00      00      GOSUBL =expexc
1A494 76BF              GOSUB  XXHEAD          Remove string header
1A498 31D0              LCHEX  0D              Carriage return
1A49C 21                P=     1
                *
                *  The following subroutine exits through EXPR but this
```

```
235                *  returns immediately since D0 is still pointing to
236                *  the token that caused the expression execution done
237                *  above to terminate.
238                *
239 1A49E 7BB0             GOSUB  STKWP          Add CR to stack
240 1A4A2 D2               C=0    A
241 1A4A4 302              LC(1)  2              Null stack length limit=2
242 1A4A7 108              R0=C
243 1A4AA 3280             LC(3)  =bSTART        STARTUP buffer ID
          8
244 1A4AF 7880             GOSUB  CHAR50
245 1A4B3 D8               B=A    A              Copy buffer length
246                *       C=0    A              (C(A)=2)
247 1A4B5 310C             LC(2)  96*2           95 chars plus CR
248 1A4B9 8BA              ?A<=C  A              Is this string too long?
249 1A4BC 95               GOYES  CHAR40         No, then okay
250 1A4BE 8C00             GOLONG =STROVF        Yes, then give string overflow
          00
251                *-
252                *-
```

```
253                     EJECT
254                 ****************************************************************
255                 ****************************************************************
256                 **
257                 ** Name:    CHARST  -  CHARSET statement execution
258                 **
259                 ** Category:   STEXEC
260                 **
261                 ** Purpose:
262                 **      Implements CHARSET statement execution
263                 **
264                 ** Entry:
265                 **      DO points past CHARSET token
266                 **      P       =  0
267                 **
268                 ** Exit:
269                 **      Exit through NXTSTM
270                 **
271                 ** Calls:      EXPEXC,XXHEAD,STKWP,I/OALL,REVPOP,I/ODAL,NXTSTM
272                 **
273                 ** Algorithm:
274                 **      Evaluate expression
275                 **      Remove string header
276                 **      Append 5 bytes of zero in case string is not a
277                 **         multiple of 6 bytes
278                 **      If null string supplied then
279                 **         Deallocate charset buffer
280                 **         Exit through NXTSTM
281                 **      If string contains more than 128 patterns supplied then
282                 **         consider string length to be 128*6
283                 **      Calculate size of buffer needed
284                 **      Allocate buffer of size needed
285                 **      If not enough memory then
286                 **         Exit and give "Insufficient Memory" error
287                 **      Move charset string into buffer
288                 **      Exit through NXTSTM
289                 **
290                 ** History:
291                 **
292                 **    Date     Programmer              Modification
293                 **   --------   ----------   ------------------------------------
294                 **  10/25/83   B.S.         Added documentation
295                 **
296                 ****************************************************************
297                 ****************************************************************
298                  ColCnt EQU    6
299
300 1A4C4 0000          REL(5) =DELAYd
          0
301 1A4C9 0000          REL(5) =STRNGP
          0
302 1A4CE 8E00 =CHARST GOSUBL =expexc
          00
303 1A4D4 767F          GOSUB  XXHEAD
304 1A4D8 AF2           C=0    W
```

```
 305 1A4DB 29           P=     9
 306             *
 307             *
 308             *  The following subroutine exits through EXPR but this
 309             *  returns immediately since D0 is still pointing to
 310             *  the token that caused the expression execution done
 311             *  above to terminate.
 312             *
 313 1A4DD 7C70          GOSUB  STKWP          Append 5 bytes of zeros
 314 1A4E1 D2            C=0    A
 315 1A4E3 30B           LC(1)  11             Null stack length limit=11
 316 1A4E6 108           R0=C                  Save null stack length limit
 317 1A4E9 32BF          LC(3)  =bCHARS
           B
 318 1A4EE 7940          GOSUB  CHAR50
 319 1A4F2 3400  CHAR10  LC(5)  128*2*ColCnt
           600
 320 1A4F9 8B2           ?A<C   A              Is it longer than 128 patterns?
 321 1A4FC 40            GOYES  CHAR20         No, then okay
 322 1A4FE DA            A=C    A              Yes, then use first 128 patterns
 323 1A500 32C0  CHAR20  LC(3)  2*ColCnt
           0
 324 1A505 D1            B=0    A
 325 1A507 A31   CHAR30  B=B+C  X
 326 1A50A B3A           A=A-C  X
 327 1A50D 59F           GONC   CHAR30         Loop back until done
 328 1A510 B31           B=B-C  X              Undo one add
 329 1A513 D4            A=B    A
 330 1A515 101   CHAR40  R1=A                  Save actual length
 331 1A518 11A           C=R2                  Recall buffer id
 332 1A51B 8F00          GOSBVL =I/OALL        Allocate buffer of desired length
           000
 333 1A522 521           GONC   CHARER         Insufficient memory?
 334 1A525 110           A=R0                  Recall string start
 335 1A528 130           D0=A                  D0 points to start of source
 336 1A52B 119           C=R1                  Recall length
 337 1A52E 8D00          GOVLNG =MOVNXT        Copy string to buffer/goto NXTSTM
           000
 338             *-
 339             *-
 340 1A535 8C00 =CHARER  GOLONG =MEMERJ        GOTO MEMERR
           00
 341             *-
 342             *-
 343             *  CHAR50:  Reverse string and check if string is considered
 344             *           null.  Returns if not null otherwise
 345             *           it deallocates buffer and exits through NXTSTM.
 346             *    Entry:  C(B)=Buf id,R0=string length considered to be null
 347             *    Exit:   R2(B)=Buf id, R0=String pointer, A(A)=String length
 348             *
 349 1A53B 10A   CHAR50 R2=C                  Save buffer number in R2
 350 1A53E 8F00          GOSBVL =REVPOP        Call REV$ and POP1S
           000
 351 1A545 137           CD1EX
 352 1A548 128           CR0EX                 Save string pointer in R0
```

```
353 1A54B 8B6          ?A>C   A              Is length < null string length?
354 1A54E 00           RTNYES                No, then return
355 1A550 11A          C=R2                  Recall buffer number
356 1A553 8E00         GOSUBL =i/odal        Yes, then deallocate buffer
          00
357 1A559 60FE         GOTO   nxtstm
358               *-
359               *-
360               ******************************************************************
361               ******************************************************************
362               **
363               ** Name:    STKWP   -  Add C(WP) & string head to math stack
364               **
365               ** Category:   MTHSTK
366               **
367               ** Purpose:
368               **      Adds P+1 nibbles of C register to top of math stack
369               **      then adds string header to stack
370               **
371               ** Entry:
372               **      D0 is expression program counter
373               **      D1 points to math stack (Where tokens are to be added)
374               **      R1(A) points to start of stack item
375               **      C(WP) = Data to be added to stack
376               **      D(A) = (AVMEMS)
377               **
378               ** Exit:
379               **      D1 updated (Original D1 + P + 1 + 16)
380               **      Exits through EXPR
381               **
382               ** Calls:      ADHEAD
383               **
384               ** Uses.......
385               **  Inclusive: See EXPR
386               **
387               ** Stk lvls:   4
388               **
389               ** History:
390               **
391               **    Date     Programmer             Modification
392               **  --------   ----------   ----------------------------------
393               **  10/25/83   B.S.         Added documentation
394               **
395               ******************************************************************
396               ******************************************************************
397 1A55D 840   =STKWP  ST=0   0              Don't return from ADHEAD
398 1A560 137          CD1EX
399 1A563 FA           C=-C   A
400 1A565 809          C+P+1
401 1A568 FA           C=-C   A
402 1A56A 8B3          ?C<D   A
403 1A56D 8C           GOYES  CHARER
404 1A56F 137          CD1EX
405 1A572 1551         DAT1=C WP
406 1A576 20           P=     0
```

```
407 1A578 6000          GOTO    =ADHDJ
408             *-
409 1A57C               END
```

ADHDj	Ext			-	407		
AVMEME	Ext			-	179		
AVMEMS	Ext			-	170		
CHAR10	Abs	107762	#1A4F2	-	319		
CHAR20	Abs	107776	#1A500	-	323	321	
CHAR30	Abs	107783	#1A507	-	325	327	
CHAR40	Abs	107797	#1A515	-	330	249	
CHAR50	Abs	107835	#1A53B	-	349	244	318
=CHARER	Abs	107829	#1A535	-	340	333	403
=CHARST	Abs	107726	#1A4CE	-	302		
ColCnt	Abs	6	#00006	-	298	319	323
D=@D1	Abs	107626	#1A46A	-	171	180	
=D=AVME	Abs	107638	#1A476	-	178		
=D=AVMS	Abs	107616	#1A460	-	169		
DELAYd	Ext			-	226	300	
DROPDC	Ext			-	55		
ENDL00	Abs	107524	#1A404	-	67	60	
ENDL10	Abs	107573	#1A435	-	80	86	
ENDL20	Abs	107594	#1A44A	-	87	81	
=ENDLIN	Abs	107492	#1A3E4	-	57		
EOLLEN	Ext			-	61	73	
I/OALL	Ext			-	332		
InvArg	Ext			-	52		
MEMERj	Ext			-	340		
MOVNXT	Ext			-	337		
OPSTRp	Ext			-	56		
POP1S	Ext			-	68	128	
REN180	Ext			-	88		
REVPOP	Ext			-	350		
=STKWP	Abs	107869	#1A55D	-	397	239	313
STRNGP	Ext			-	227	301	
STROVF	Ext			-	250		
=STRTUP	Abs	107662	#1A48E	-	228		
=XXHEAD	Abs	107598	#1A44E	-	128	230	303
bCHARS	Abs	3067	#00BFB	-	11	317	
bSTART	Abs	2056	#00808	-	11	243	
expexc	Ext			-	67	229	302
i/odal	Ext			-	356		
invarg	Abs	107476	#1A3D4	-	52	72	
nxtstm	Abs	107594	#1A44A	-	88	64	357

Input Parameters

  Source file name is SB&EXC::MS

  Listing file name is SB/EXC:TI:ML::-1

  Object file name is SB%EXC:TI:MS::-1

```
                                  111111
                         0123456789012345
```
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *       SSS    CCC    &     RRRR   EEEEE  N   N
 2            *      S   S  C   C  & &    R   R  E      N   N
 3            *      S      C      & &    R   R  E      NN  N
 4            *       SSS   C       &     RRRR   EEEE   N N N
 5            *          S  C      & & &  R R    E      N  NN
 6            *      S   S  C   C  &  &   R  R   E      N   N
 7            *       SSS    CCC    && &  R   R  EEEEE  N   N
 8            *
 9            *
10            *
11
12                   TITLE  RENUMBER Module <831216.1609>
13 1A57C             ABS    #1A57C
```

```
14                      EJECT
15                *
16                **********************************************************************
17                **********************************************************************
18                **
19                ** Name:    RENUM   -  Renumber Execution
20                **
21                ** Category: STEXEC
22                **
23                ** Purpose: Renumber partial or all the program line in a file.
24                **
25                ** Entry: D0 points at past the RENUM token
26                **
27                ** Exit: Exit to NXTSTM
28                **
29                ** Calls: CHKPSF, PRSCOP, RENSUB, UPDCRL, GETLN#, FINDL, LINE#1,
30                **        CPL#10
31                **
32                ** Uses: A-D, R0-R3, D0,D1, S0-2
33                **
34                ** Stk lvl: 4
35                **
36                ** History:
37                **
38                **    Date      Programmer    Modification
39                **  --------    ----------    ------------
40                **  03/01/83    JP            Packed GETSTC/GETPRO to CHKPSF
41                **
42                **********************************************************************
43                **********************************************************************
44                *
45 1A57C 0000          REL(5) =RNUMDC
          0
46 1A581 0000          REL(5) =RENMP
          0
47 1A586 136   =RENUM  CDOEX                 SAVE D0 IN R0
48 1A589 108           R0=C
49                *
50                * Check File type
51                *   Error Exit if non BASIC
52                * Check Protection
53                *   Error Exit if PRIVATE  or SECURE
54                *
55 1A58C 8F00          GOSBVL =CHKPSF
          000
56                *
57                * Chain the subprograms and labels if need to
58                *
59 1A593 8F00          GOSBVL =PRSCOP
          000
60                *
61                * FIRST WE DO IS TO COMPILE ALL THE LINE# REFERENCE
62                *
63 1A59A 851    RENO05 ST=1   1
64 1A59D 852           ST=1   2
```

```
 65 1A5A0 7FA1          GOSUB  RENSUB          Compile all references
 66 1A5A4 4C1           GOC    REN010          NO ERROR IN COMPILE
 67 1A5A7 1F00          D1=(5) =PCADDR
          000
 68 1A5AE 11A           C=R2
 69 1A5B1 145           DAT1=C A
 70 1A5B4 8F00          GOSBVL =UPDCRL
          000
 71 1A5BB 8C00          GOLONG =STMTNF
          00
 72               *
 73               *  WHILE WE ARE ASSIGNING NEW LINE #, R0-R3 CONTAIN :
 74               *   R0 = BEGINNING LINE # ( SPECIFIED BY USER OR DEFAULT TO 10)
 75               *   R1 = INCREMENT
 76               *   R2 = POINTER POINTS TO THE 1ST LINE WOULD BE RENUMBERED
 77               *   R3 = LAST LINE# WOULD BE RENUMBERED
 78               *   D  = POINTER OF END OF FILE
 79               *   B  = INCREMENT
 80               *   D1 = LINE # POINTER
 81               *
 82 1A5C1 110    REN010 A=R0
 83 1A5C4 130           D0=A                   RESTORE D0
 84 1A5C7 72C3          GOSUB  LINE#1          GET POINTER OF 1ST LINE #
 85 1A5CB 102           R2=A                   Save ptr to line#
 86 1A5CE D2            C=0    A
 87 1A5D0 E6            C=C+1  A
 88 1A5D2 F2            CSL    A               C(A) = 00010
 89 1A5D4 108           R0=C                   SET DEFAULT STARTING LINE #
 90 1A5D7 109           R1=C                   SET DEFAULT INCREMENT
 91 1A5DA F2            CSL    A
 92 1A5DC F2            CSL    A
 93 1A5DE F2            CSL    A               C(A) = 10000
 94               *
 95 1A5E0 10B           R3=C                   SET DEFAULT LAST LINE #
 96               * NOW START TO PROCESS THE ARGUMENTS
 97 1A5E3 14A           A=DAT0 B
 98 1A5E6 7821          GOSUB  GETLN#
 99 1A5EA 436           GOC    REN100          NO ARGUMENT AT ALL
100 1A5ED 100           R0=A
101 1A5F0 7E11          GOSUB  GETLN#
102 1A5F4 495           GOC    REN100          NO INCREMENT
103 1A5F7 101           R1=A
104 1A5FA 7411          GOSUB  GETLN#
105 1A5FE 4F4           GOC    REN100
106 1A601 D6            C=A    A               Start line# in C(A)
107 1A603 132           ADOEX                  SAVE D0 IN R2
108 1A606 102           R2=A
109 1A609 8F00          GOSBVL =FINDL
          000
110 1A610 860           ?ST=0  0
111 1A613 60            GOYES  REN020          FOUND GREATER LINE #
112 1A615 66F0          GOTO   REN180          DO NOTHING IF LINE# NOT FOUND
113 1A619 137   REN020  CD1EX                  SAVE STARTING POINTER IN R2
114 1A61C 12A           CR2EX
115 1A61F 134           D0=C                   RESTORE D0
```

```
116 1A622 7CE0          GOSUB  GETLN#
117 1A626 472           GOC    REN100
118 1A629 05            SETDEC                  INCREMENT THE LINE # BY 1
119 1A62B 23            P=     3
120 1A62D B14           A=A+1  WP
121 1A630 20            P=     0
122 1A632 04            SETHEX
123 1A634 491           GOC    REN100
124                *
125                * Find a line number is larger than the specified one.
126                *
127 1A637 D6            C=A    A
128 1A639 8F00          GOSBVL =FINDL
            000
129 1A640 D0            A=0    A
130 1A642 15B3          A=DAT1 4                Read the next line number
131 1A646 870           ?ST=1  0                Line # not found ?
132 1A649 50            GOYES  REN100           If so, last line is 10000
133 1A64B 103           R3=A                    Ending line# +1
134                *
135 1A64E 7B33  REN100  GOSUB  LINE#1           GET 1ST LINE # ADDRESS
136 1A652 137           CD1EX
137 1A655 112           A=R2                    A= START LINE # ADDRESS
138 1A658 8A2           ?A=C   A                START FROM 1ST LINE ?
139 1A65B 32            GOYES  REN105           IF SO, NO NEED TO CHECK
140 1A65D 130           D0=A
141 1A660 183           D0=D0- 4                POINTS INTO PREVIOUS LINE
142 1A663 8F00          GOSBVL =CPL#10          GET PREVIOUS LINE #
            000
143 1A66A D2            C=0    A
144 1A66C 15F3          C=DAT1 4                C = PREVIOUS LINE #
145 1A670 110           A=R0
146 1A673 8B6           ?A>C   A                START LINE# > PREVIOUS LINE#
147 1A676 80            GOYES  REN105           IF SO, OK
148 1A678 8C00          GOLONG =InvArg          OTHERWISE, SAY "ARG. OUT OF RANGE"
            00
149                *
150 1A67E 1F00  REN105  D1=(5) =CURREN
            000
151 1A685 147           C=DAT1 A
152 1A688 D7            D=C    A                D= POINTER OF END OF FILE
153 1A68A 11A           C=R2
154 1A68D 135           D1=C
155 1A690 D2            C=0    A
156 1A692 15F3          C=DAT1 4
157 1A696 134           D0=C                    SAVE START LINE # IN  D0
158 1A699 11A   REN110  C=R2
159 1A69C 135           D1=C                    POINTS TO 1ST LINE #
160 1A69F 111           A=R1
161 1A6A2 D8            B=A    A                B= INCREMENT
162                *
163                * D0 = START RENUMBER LINE # POINTED BY R2
164                * D1 POINTS AT LINE #
165                * R0 = NEXT LINE # TO BE ASSIGNED
166                * R1 = INCREMENT
```

```
167                 * R2 = POINTER OF 1ST LINE NUMBER TO BE RENUMBERED
168                 * R3 = LAST LINE # TO BE RENUMBERED
169                 *
170 1A6A4 137    REN120 CD1EX
171 1A6A7 8BF           ?C>=D  A              AT END OF FILE YET ?
172 1A6AA 85            GOYES  REN170         IF SO, GOTO REN170
173 1A6AC 135           D1=C
174 1A6AF D0            A=0    A
175 1A6B1 15B3          A=DAT1 4              READ THE OLD LINE # FIRST
176 1A6B5 11B           C=R3                  SEE IF WE PASS LAST LINE # YET
177 1A6B8 8BE           ?A>=C  A
178 1A6BB 74            GOYES  REN170         IF SO, WE ARE DONE
179 1A6BD 110           A=R0                  Next line# to assign
180 1A6C0 8B2           ?A<C   A              WE BOMB ?
181 1A6C3 41            GOYES  REN140         IF NOT, GOTO REN140
182 1A6C5 04     REN130 SETHEX
183 1A6C7 136           CD0EX                 GET RESTART LINE #
184 1A6CA 108           R0=C
185 1A6CD D2            C=0    A              SET INCREMENT TO 1
186 1A6CF E6            C=C+1  A              rest of block to be
187 1A6D1 109           R1=C                  incremented by 1
188 1A6D4 54C           GONC   REN110         (B.E.T)
189 1A6D7 1593   REN140 DAT1=A 4
190 1A6DB 05            SETDEC
191 1A6DD C0            A=A+B  A
192 1A6DF 04            SETHEX
193 1A6E1 100           R0=A                  Next line# to assign
194 1A6E4 173           D1=D1+ 4              PASS LINE#, POINTS AT LINE LENGTH
195 1A6E7 D2            C=0    A
196 1A6E9 14F    REN160 C=DAT1 B              READ THE LINE LENGTH
197 1A6EC 133           AD1EX
198 1A6EF CA            A=A+C  A              POINTS TO END OF LINE
199 1A6F1 131           D1=A                  D1 at t@ or tEOL
200 1A6F4 14B           A=DAT1 B
201 1A6F7 171           D1=D1+ 2
202 1A6FA 908           ?A=0   P              POINTS AT EOL ?
203 1A6FD 7A            GOYES  REN120
204 1A6FF 59E           GONC   REN160         (B.E.T.)
205                 *
206                 *  THIRD STEP IS TO DECOMPILE THE LINE #
207                 *
208 1A702 851    REN170 ST=1   1
209 1A705 842           ST=0   2
210 1A708 7740          GOSUB  RENSUB         Renumber line# references
211 1A70C 8C00 =REN180 GOLONG =NXtstm
            00
212                 *
213                 ********************************************************************
214 1A712 14A    GETLN# A=DAT0 B
215 1A715 161           D0=D0+ 2
216 1A718 3100          LC(2)  =tCOMMA
217 1A71C 966           ?A#C   B
218 1A71F 00            RTNYES                NO MORE ARGUMENT
219 1A721 D0            A=0    A
220 1A723 15A3          A=DAT0 4
```

```
221 1A727 163          D0=D0+ 4
222 1A72A 03           RTNCC
223             *
224             ******************************************************************
225             ******************************************************************
226             **
227             ** Name:(S) RENSUB  -  Renumber Subroutine
228             **
229             ** Category:   FILUTL
230             **
231             ** Purpose : 1. Compile all line number references
232             **           2. Clear all compiled offsets
233             **           3. Renumber all line number references
234             **
235             ** Entry : CURRST & CURREN pts current file
236             **         S1 = 0 - Only clear compiled offset
237             **            = 1 - Compile offset or renumber line number
238             **         If S1=1 :
239             **            S2 = 1 - Compile reference offset
240             **            S2 = 0 - Renumber line number
241             **
242             ** Exit : Carry set => No error
243             **        Carry clear=> Line number not found
244             **                      R2= ptr to stmt len of stmt in error
245             **
246             ** Calls : PFNDL*, EXPSKP, FINDA, ISRAM?, LINE#1, POLL
247             **
248             ** Uses : A,B(A),C,D(A),D0,D1,R2, S3
249             **
250             ** Stk lvls : +2
251             **
252             ** Detail :
253             **
254             ** The line number is expected to be found in the following
255             ** mainframe statements:
256             **
257             **   1. GOTO/GOSUB/RESTORE <LINE#/LABEL>
258             **   2. ON ERROR GOTO/GOSUB <LINE#/LABEL>
259             **   3. ON TIMER [#<exp>,] <exp> GOTO/GOSUB <LINE#/LABEL>
260             **   4. ON <exp> GOTO/GOSUB/RESTORE <LINE#/LABEL>,...
261             **   5. IF <exp> THEN LINE#/LABEL/EXT [ELSE LINE#/LABEL/EXT IF]
262             **   6. PRINT USING LINE#/LABEL
263             **   7. DISP USING LINE#/LABEL
264             **   8. ON INTR GOTO/GOSUB LINE#/LABEL
265             **   9. POLL for non-mainframe XWORD
266             **
267             ** For XWORD (External) statements, the line number is handled
268             ** as follows:
269             **
270             ** . If RENSUB is just called for zeroing the compiled offset
271             **   (S1=0), the line # in XWORD statement will be ignored. This
272             **   means the execution of an XWORD statement has to assume the
273             **   compiled offset is incorrect and has to zero it everytime.
274             ** . If RENSUB is called for renumbering (S1=1), the poll pREN
275             **   will be issued so that each LEX file that contains XWORD
```

```
276             **   statements that may have line numbers will be allowed to
277             **   supply the correct renumbering.  See the pREN poll inter-
278             **   face for details.
279             **
280             ********************************************************************
281             ********************************************************************
282             **
283             ********************************************************************
284             ********************************************************************
285             **
286             ** Name:(S) pREN    -  Renumber an XWORD line# reference
287             **
288             ** Category:   POLL
289             **
290             ** Type:       POLL
291             **
292             ** Purpose:
293             **      Renumber a XWORD statement if it has line number as its
294             **      arguments.
295             **
296             ** Should poll be "Handled" (return with XM=0)?:Yes
297             **
298             ** Meaning of "Handling" Poll (what does code do if handled?):
299             **      Return D1 points to where the line number is.
300             **
301             ** Entry conditions for handler (registers, ST, RAM, etc.):
302             **      B[A] = Poll number.
303             **      HEX mode.
304             **      P=0.
305             **      A[4-0] = LEX file ID and fcn #
306             **      D1 past the XWORD tokens.
307             **
308             ** Normal exit conditions from handler if handled (ST, RAM,
309             ** registers, etc.):
310             **      Carry clear (POLL only).
311             **      HEX mode.
312             **      XM=0.
313             **      D1 @ the line number token(tLINE# or tLITRL)
314             **      S3 = 1, if there are more than one line numbers followed.
315             **
316             ** Normal exit conditions from handler if not handled (ST, RAM,
317             ** registers, etc.):
318             **      Carry clear.
319             **      HEX mode.
320             **      XM=1.
321             **
322             ** Error exit conditions from handler (POLL only):
323             **      Carry set.
324             **      HEX mode.
325             **      Will exit to MEMERR(Insufficient Memory).
326             **
327             ** Available subroutine levels: 5
328             **
329             ** What registers/RAM may be used if handled?:
330             **      A-D, D0, P
```

```
331             **
332             ** What registers/RAM may be used if not handled?:
333             **      A-D, D0, P
334             **
335             ** What registers/RAM may be used if error exit (POLL only)?:
336             **      A-D, D0, P
337             **
338             ** History:
339             **
340             **    Date     Programmer             Modification
341             **  --------   ----------   ----------------------------------
342             **  04/20/83   SC           Document
343             **
344             ********************************************************************
345             ********************************************************************
346             *
```

```
347                     STITLE CLLINK - Clear Links, Compile Line#s
348                  ****************************************************************
349                  ****************************************************************
350                  **
351                  ** Name:    CLLINK  -  Clear Links, Compile Line Numbers
352                  **
353                  ** Category:   FILUTL
354                  **
355                  ** Purpose:
356                  **      Clear Sub links, Label Links,
357                  **      Compiled offsets of all line# in the current file
358                  **
359                  ** Entry:
360                  **      P      =  0
361                  **
362                  ** Exit:
363                  **      Through RENSUB
364                  **
365                  ** Calls:      None
366                  **
367                  ** Uses.......
368                  **  Exclusive: A,B,C,D,D0,D1,R2,S1
369                  **
370                  ** Stk lvls:   1
371                  **
372                  ** NOTE:      (remove any item below this point if not needed)
373                  **      If Sublink = 0
374                  **         Nothing is done
375                  **      else
376                  **         Both Sub-program and Label links are zeroed
377                  **         All line# offsets are zeroed
378                  **
379                  ** Detail:     Must precede RENSUB
380                  **
381                  ** History:
382                  **
383                  **    Date     Programmer             Modification
384                  **  --------   ----------   ---------------------------------
385                  **  06/30/82   JP           Modified documentation
386                  **
387                  ****************************************************************
388                  ****************************************************************
389                  *
390 1A72C 1F00 =CLLINK D1=(5) =CURRST
          000
391 1A733 143           A=DAT1 A
392 1A736 D2            C=0    A
393 1A738 3100          LC(2)  =oSUBLn       LOAD OFFSET FROM CURRST TO SUBLNK
394 1A73C CA            A=A+C  A
395 1A73E 131           D1=A
396 1A741 143           A=DAT1 A
397 1A744 8A8           ?A=0   A
398 1A747 00            RTNYES
399 1A749 AF2           C=0    W
400 1A74C 15D9          DAT1=C 10
```

```
401 1A750 841          ST=0   1
402             *
403 1A753 04   =RENSUB SETHEX
404 1A755 1B00         DO=(5) =CURREN
          000
405 1A75C 146          C=DAT0 A            C= CURREN
406             *
407 1A75F CE           C=C-1  A            CHECK IF CURRENT FILE IS IN RAM
408 1A761 8F00         GOSBVL =ISRAM?       OR IRAM ?
          000
409 1A768 4C0          GOC    RNS150
410 1A76B 3100         LC(2)  =eFACCS      IF NOT, SAY "ILLEGAL ACCESS"
411 1A76F 8C00 =MFerrj GOLONG =MFERRj
          00
412 1A775 7412  RNS150 GOSUB  LINE#1        GET 1ST LINE # ADDRESS
413 1A779 6E91         GOTO   RNS630
414 1A77D 137   RNS160 CD1EX
415 1A780 10A          R2=C                 R2 = PTS AT CURRENT STMT LEN BYTE
416 1A783 135          D1=C
417 1A786 171          D1=D1+ 2             PAST STMT LENGTH BYTE
418 1A789 843   RNS170 ST=0   3             CLEAR ON <EXP> FLAG
419 1A78C 14B          A=DAT1 B             READ BEGIN TOKEN
420 1A78F 171          D1=D1+ 2             PASS BEGIN BASIC TOKEN
421 1A792 8E00         GOSUBL =FINDAj
          00
422 1A798 00           CON(2) =tGOSUB
423 1A79A D21          REL(3) RNS600        GOSUB
424 1A79D 00           CON(2) =tGOTO
425 1A79F 821          REL(3) RNS600        GOTO
426 1A7A2 00           CON(2) =tRESTR
427 1A7A4 321          REL(3) RNS600        RESTORE
428 1A7A7 00           CON(2) =tON
429 1A7A9 120          REL(3) RNS300        ON
430 1A7AC 00           CON(2) =tPRINT
431 1A7AE 350          REL(3) RNS400        PRINT
432 1A7B1 00           CON(2) =tDISP
433 1A7B3 E40          REL(3) RNS400        DISP
434 1A7B6 00           CON(2) =tIF
435 1A7B8 F60          REL(3) RNS500        IF
436 1A7BB 00           CON(2) =tELSE
437 1A7BD 170          REL(3) RNS510        ELSE
438 1A7C0 00           CON(2) =tXWORD
439 1A7C2 F70          REL(3) RNS520        XWORD
440 1A7C5 00           CON(2) 0
441 1A7C7 5B5          GONC   RNS410        (B.E.T.)
442             *
443 1A7CA 14B   RNS300 A=DAT1 B
444 1A7CD 3100         LC(2)  =tERROR
445 1A7D1 962          ?A=C   B
446 1A7D4 31           GOYES  RNS320        ON ERROR
447 1A7D6 E6           C=C+1  A
448 1A7D8 962          ?A=C   B
449 1A7DB 61           GOYES  RNS350        ON TIMER
450             * ON <exp> GOTO/GOSUB/RESTORE <LINE#/LABEL>[,LINE#/LABEL][,..]
451 1A7DD 853          ST=1   3             MUST BE ON <exp>
```

```
452 1A7E0 78C1  RNS310 GOSUB  EXPSKP          SKIP OVER EXPRESSION
453 1A7E4 450          GOC    RNS330          (B.E.T.)
454            * ON ERROR GOTO/GOSUB <LINE#/LABEL>
455 1A7E7 171   RNS320 D1=D1+ 2               Step over tokens
456 1A7EA 171   RNS330 D1=D1+ 2
457 1A7ED 69D0  RNS340 GOTO   RNS600
458            * ON TIMER [#<exp>,] <exp> GOTO/GOSUB
459            *
460 1A7F1 171   RNS350 D1=D1+ 2               Step over tTIMER
461 1A7F4 74B1         GOSUB  EXPSKP
462 1A7F8 171          D1=D1+ 2               Step over tCOMMA
463 1A7FB 54E          GONC   RNS310          (B.E.T.)
464            *
465            * OUTPUT/ENTER [USING]
466            *
467 1A7FE 173   RNS390 D1=D1+ 4
468            *
469            * PRINT/DISP USING LINE#/LABEL
470 1A801 14B   RNS400 A=DAT1 B
471 1A804 3100         LC(2)  =tUSING
472 1A808 966          ?A#C   B
473 1A80B 81           GOYES  RNS410
474            *
475            * See if USING followed by a line number token (tLINE#)
476            *
477 1A80D 171          D1=D1+ 2
478 1A810 14B          A=DAT1 B
479 1A813 171          D1=D1+ 2
480 1A816 3100         LC(2)  =tLINE#
481 1A81A 966          ?A#C   B
482 1A81D 60           GOYES  RNS410
483            *
484 1A81F 6411         GOTO   RNS650
485            *
486 1A823 6460  RNS410 GOTO   RNS540          (B.E.T.)
487            *
488            * IF <exp> THEN LINE#/LABEL/EXT IF
489            *
490 1A827 7181  RNS500 GOSUB  EXPSKP
491 1A82B 171          D1=D1+ 2               Step over tTHEN
492 1A82E 14B   RNS510 A=DAT1 B
493 1A831 3100         LC(2)  =tEXTIF
494 1A835 966          ?A#C   B
495 1A838 5B           GOYES  RNS340
496 1A83A 171          D1=D1+ 2               PASS tEXTIF
497 1A83D 6F3F         GOTO   RNS160          D1 at Stmt Length Byte
498            *
499            * XWORD STATEMENT
500            *
501 1A841 D0    RNS520 A=0    A
502 1A843 15B3         A=DAT1 4               READ LEX FILE ID AND FCN #
503 1A847 173          D1=D1+ 4               SKIP OVER XWORD TOKENS
504 1A84A 3110         LC(2)  01
505 1A84E 962          ?A=C   B               IS THIS A MAINFRAME XWORD ?
506 1A851 73           GOYES  RNS540          IF SO, NO LINE #
```

```
507 1A853 3400          LC(5)  =tINTR          ON INTR ? (#15FF)
          000
508 1A85A 8A2           ?A=C   A
509 1A85D D8            GOYES  RNS330          IF SO, GOTO RNS330
510 1A85F A2E           C=C-1  XS
511 1A862 8A2           ?A=C   A               ENTER ? (#14FF)
512 1A865 72            GOYES  RNS550
513 1A867 A2E           C=C-1  XS
514 1A86A 8A2           ?A=C   A               OUTPUT ? (#13FF)
515 1A86D F1            GOYES  RNS550
516 1A86F 861           ?ST=0  1               JUST FOR ZEROING OFFSET ?
517 1A872 61            GOYES  RNS540
518 1A874 8E00          GOSUBL =POlljj
          00
519 1A87A 00            CON(2) =pREN
520 1A87C 560           GONC   RNS530
521 1A87F 6000          GOTO   =CHARER         GOTO MEMERR
522               *
523 1A883 831   RNS530 ?XM=0
524 1A886 14            GOYES  RNS600          Handled without error?
525 1A888 6470  RNS540 GOTO   RNS620
526               *
527               * ENTER & OUTPUT have device specifier followed.
528               * Here is try to skip over the device specifier.
529               * Following tokens could follow the ENTER or OUTPUT
530               *
531               * 1)    <string expression> <t@>
532               * 2) or <tCOLON> <num expr> [ <tCOMMA> <num expr> ] <t@>
533               * 3) or <tCOLON> <tLITRL> <device word> [ [<tCOLON> <num expr>]
534               *       <tCOMMA> <num expr> ] ] <t@>
535               * 4) or <tCOLON> <t%> <num expr> [ [<tCOLON> <num expr> ]
536               *       <tCOMMA> <num expr> ] ] <t@>
537               * 5) or <tCOLON> <tLITRL> <assign word> <t@>
538               * 6) or <tCOLON> <tLITRL> <device ID> [ [<tCOLON> <num expr>]
539               *       <tCOMMA> <num expr> ] ] <t@>
540               *
541               *
542 1A88C 14B   RNS550 A=DAT1 B
543 1A88F 3100          LC(2)  =tCOLON
544 1A893 966           ?A#C   B
545 1A896 D1            GOYES  RNS570          If not COLON, is string expr.
546 1A898 171           D1=D1+ 2
547 1A89B 14B           A=DAT1 B
548 1A89E 3100          LC(2)  =tLITRL
549 1A8A2 962           ?A=C   B
550 1A8A5 B0            GOYES  RNS560
551 1A8A7 3100          LC(2)  =t%
552 1A8AB 966           ?A#C   B
553 1A8AE 50            GOYES  RNS570
554               *
555 1A8B0 171   RNS560 D1=D1+ 2
556               *
557 1A8B3 75F0  RNS570 GOSUB  EXPSKP
558               *
559 1A8B7 3100          LC(2)  =t@
```

```
560 1A8BB 966          ?A#C   B
561 1A8BE 2F           GOYES  RNS560
562 1A8C0 171          D1=D1+ 2              Skip over t@
563              *
564 1A8C3 6D3F         GOTO   RNS400
565              *
566              *
567              * GOTO/GOSUB/RESTORE <LINE#/LABEL>
568              *
569              * 1). NOW D1 POINTS PAST GOTO/GOSUB/RESTORE TOKEN.
570              *     R2 POINTS TO THE LINE LENGTH OF CURRENT STATEMENT.
571              * 2). IF D1 POINTS AT A tLINE# TOKEN, THEN 5 NIBS OF LINK FIELD
572              *     AND 4 NIBS OF LINE NUMBER WILL FOLLOW.
573              *     IF D1 POINTS AT A tLBLRF TOKEN, THEN IT MAY BE FOLLOWED BY
574              *     THE tLITRL TOKEN OR STRING EXPRESSION
575              *
576 1A8C7 14B   RNS600 A=DAT1 B
577 1A8CA 171          D1=D1+ 2
578 1A8CD 3100         LC(2)  =tLINE#
579 1A8D1 962          ?A=C   B
580 1A8D4 06           GOYES  RNS650
581 1A8D6 3100         LC(2)  =tLITRL
582 1A8DA 14B          A=DAT1 B
583 1A8DD 966          ?A#C   B              STRING EXPRESSION FOLLOWS?
584 1A8E0 50           GOYES  RNS610
585 1A8E2 171          D1=D1+ 2              PASS tLITRL TOKEN
586 1A8E5 73C0  RNS610 GOSUB  EXPSKP
587 1A8E9 863   RNS615 ?ST=0  3              "ON" FLAG SET ?
588 1A8EC 11           GOYES  RNS620         IF NOT, GOTO NEXT STATEMENT
589 1A8EE 14B          A=DAT1 B
590 1A8F1 171          D1=D1+ 2
591 1A8F4 3100         LC(2)  =tCOMMA
592 1A8F8 962          ?A=C   B
593 1A8FB CC           GOYES  RNS600
594              *
595              * MOVE D1 TO END OF CURRENT STATEMENT AND CONTINUE
596              *
597 1A8FD 112   RNS620 A=R2
598 1A900 131          D1=A
599 1A903 D2           C=0    A
600 1A905 14F          C=DAT1 B
601 1A908 CA           A=A+C  A
602 1A90A 131          D1=A
603 1A90D 14B          A=DAT1 B
604 1A910 171          D1=D1+ 2
605 1A913 90C          ?A#0   P              AT EOL ?
606 1A916 A1           GOYES  RNS635         IF NOT, GOTO RNS640
607 1A918 173   RNS630 D1=D1+ 4              D1 POINTS AT LINE LENGTH
608 1A91B 1B00         D0=(5) =CURREN
          000
609 1A922 142          A=DAT0 A              CURREN
610 1A925 137          CD1EX
611 1A928 135          D1=C
612 1A92B 8BA          ?C>=A  A              PAST END OF FILE ?
613 1A92E 00           RTNYES                IF SO, WE ARE DONE
```

```
614 1A930 6C4E  RNS635 GOTO   RNS160
615             *
616             * FOUND A LINE NUMBER !! LET'S GET TO WORK
617             *
618 1A934 D2    RNS650 C=0    A
619 1A936 871          ?ST=1  1              DO WE WANT TO COMPILE ?
620 1A939 B0           GOYES  RNS660
621 1A93B 145          DAT1=C A              No, Clear offset
622 1A93E 178   RNS655 D1=D1+ 9              Step over line# & 5 nib field
623 1A941 57A          GONC   RNS615         (B.E.T.)
624 1A944 137   RNS660 CD1EX
625 1A947 135          D1=C                  Ptr to line#
626 1A94A 134          D0=C
627 1A94D 862          ?ST=0  2              Renumber references?
628 1A950 42           GOYES  RNS680
629             * COMPILE LINE #
630 1A952 174          D1=D1+ 5              Point past 5 nib field
631 1A955 D2           C=0    A
632 1A957 15F3         C=DAT1 4              Read in line#
633 1A95B D5           B=C    A              B(A) = LINE #
634 1A95D 7C20         GOSUB  LINE#1         GET 1ST LINE# ADDRESS
635 1A961 8F00         GOSBVL =PFNDL*        Find line# & write out offset
          000
636 1A968 500          RTNNC                 NOT FOUND
637 1A96B 136          CD0EX
638 1A96E 135          D1=C
639 1A971 4CC          GOC    RNS655         (B.E.T.)
640             * Renumber line# references
641 1A974 143   RNS680 A=DAT1 A              READ THE OFFSET
642 1A977 CA           A=A+C  A              POINTS TO LINE #
643 1A979 130          D0=A
644 1A97C 15A3         A=DAT0 4              Read in line#
645 1A980 174          D1=D1+ 5              Step over offset
646 1A983 1593         DAT1=A 4              Write out new line#
647 1A987 135          D1=C
648 1A98A 53B          GONC   RNS655         (B.E.T.)
649             *
650 1A98D 1F00 =LINE#1 D1=(5) =CURREN
          000
651 1A994 147          C=DAT1 A
652 1A997 D7           D=C    A
653 1A999 1CE          D1=D1- 15
654 1A99C 143          A=DAT1 A
655 1A99F D2           C=0    A
656 1A9A1 3100         LC(2)  =oFLSTr
657 1A9A5 CA           A=A+C  A
658 1A9A7 131          D1=A                  D1 POINTS AT 1ST LINE #
659 1A9AA 01           RTN
660             *
661             ********************************************************************
662             ********************************************************************
663             **
664             ** Name:(S) EXPSKP  -  Skip Over Tokenized Expression
665             **
666             ** Category:   FILUTL
```

```
667             **
668             ** Purpose:  Skip over tokenized expression
669             **
670             ** Entry:    D1 = Start of expression
671             **
672             ** Exit:     A = NEXT TOKEN after expression
673             **           D1= Points to next token after expression
674             **           Carry set
675             **
676             ** Calls : FINDA
677             **
678             ** Uses:     A(A) ,C(5:0), D1, S10
679             **
680             ** Stk lvls : 1
681             ********************************************************************
682             ********************************************************************
683             *
684 1A9AC 84A  =EXPSKP ST=0   10              CLEAR SKIP PARM COUNT FLAG
685 1A9AF D0    EXPS10 A=0    A
686 1A9B1 14B          A=DAT1 B              READ NEXT TOKEN
687 1A9B4 3100         LC(2)  ="
688 1A9B8 9EE          ?A>=C  B              NUMBER TOKEN ?
689 1A9BB B2           GOYES  EXPS20         IF NOT, GOTO EXPS20
690 1A9BD 171          D1=D1+ 2
691 1A9C0 303          LC(1)  3              C(0) = 3
692 1A9C3 A02          C=A+C  P
693 1A9C6 B8A          C=-C   P
694 1A9C9 80F0         CPEX   0
695 1A9CD 137          CD1EX
696 1A9D0 809          C+P+1
697 1A9D3 137          CD1EX                 Step over number
698 1A9D6 20           P=     0
699 1A9D8 BE4          ASR    B
700 1A9DB 908          ?A=0   P              INTEGER CONSTANT ?
701 1A9DE 1D           GOYES  EXPS10         IF SO, NEXT TOKEN
702 1A9E0 172          D1=D1+ 3              SKIP OVER EXP FIELD
703 1A9E3 5BC          GONC   EXPS10
704 1A9E6 31D2 EXPS20  LCHEX  2D
705 1A9EA 9EE          ?A>=C  B              Not operator or number token?
706 1A9ED 13           GOYES  EXPS30
707 1A9EF 171          D1=D1+ 2
708 1A9F2 3100         LC(2)  ="             C(B) = HEX 22
709 1A9F6 962          ?A=C   B
710 1A9F9 B0           GOYES  EXPS25
711 1A9FB 3100         LC(2)  ='             C(B) = HEX 27
712 1A9FF 966          ?A#C   B
713 1AA02 DA           GOYES  EXPS10
714             * FOUND AN SINGLE QUOTE OR DOUBLE QUOTE MARK
715             * LOOK FOR A PAIR
716 1AA04 14F   EXPS25 C=DAT1 B
717 1AA07 171          D1=D1+ 2
718 1AA0A 966          ?A#C   B
719 1AA0D 7F           GOYES  EXPS25
720 1AA0F 5F9          GONC   EXPS10         (B.E.T.)
721 1AA12 173   EXPS70 D1=D1+ 4
```

```
722 1AA15 170   EXPJ05 D1=D1+ 1
723             *
724 1AA18 171   EXPJ10 D1=D1+ 2
725 1AA1B 539          GONC   EXPS10          (B.E.T.)
726             * Not operator or number token
727 1AA1E 3104  EXPS30 LCHEX  40
728 1AA22 9EA          ?A<=C  B               single or multi-digit integer?
729 1AA25 3F           GOYES  EXPJ10
730 1AA27 3100         LC(2)  =tZ             C(B) = 5A
731 1AA2B 9E6          ?A>C   B               Not a 1-byte var name?
732 1AA2E E0           GOYES  EXPS40          Yes =>  GOTO EXPS40
733 1AA30 86A          ?ST=0  10              NO NEED TO JUMP OVER PARAM COUNT ?
734 1AA33 5E           GOYES  EXPJ10         Step over 1-byte var name
735 1AA35 172          D1=D1+ 3              Step over 1-byte var name & parm ct
736 1AA38 637F         GOTO   EXPSKP
737 1AA3C 3100  EXPS40 LC(2)  (=tADIG9)+1
738 1AA40 9E2          ?A<C   B               Not a FUNCTION ?
739 1AA43 5D           GOYES  EXPJ10          Must be 2-byte var name
740 1AA45 3100         LC(2)  =tRELOP
741 1AA49 962          ?A=C   B               Relops followed by 1 nib parm ct.
742 1AA4C 9C           GOYES  EXPJ05
743 1AA4E 3100         LC(2)  =LASTFN
744 1AA52 9E6          ?A>C   B               COMMAND ?
745 1AA55 00           RTNYES                 IF SO, WE ARE DONE
746 1AA57 8E00         GOSUBL =FINDAj
        00
747 1AA5D 00           CON(2) =LASTFN
748 1AA5F E20          REL(3) EXPS80
749 1AA62 00           CON(2) =tXFN
750 1AA64 EAF          REL(3) EXPS70
751 1AA67 00           CON(2) =tFN
752 1AA69 E10          REL(3) EXPS60
753 1AA6C 00           CON(2) =tARRAY
754 1AA6E 910          REL(3) EXPS60
755 1AA71 00           CON(2) =tDMYAR
756 1AA73 410          REL(3) EXPS60
757 1AA76 00           CON(2) =tISUB$
758 1AA78 900          REL(3) EXPS55
759             *
760 1AA7B 00           CON(2) 0
761 1AA7D 6A9F  EXPS50 GOTO   EXPJ10
762             *
763 1AA81 170   EXPS55 D1=D1+ 1
764 1AA84 58F          GONC   EXPS50          (B.E.T.)
765             *
766             * FOR ARRAYS, DUMMY ARRAYS, & FUNCTIONS, THERE IS A SUBSCRIPT
767             * COUNT AFTER THE NAME. SO WE SET A FLAG HERE, NEXT TIME WHEN
768             * SEE A VARIABLE NAME WE WILL HAVE TO JUMP OVER ONE MORE NIBBLE.
769 1AA87 85A   EXPS60 ST=1   10
770 1AA8A 52F          GONC   EXPS50           (B.E.T.)
771             *
772             * tFFN - FUNNY FUNCTION
773             * tFFN dd dd len .....
774             *
775 1AA8D 175   EXPS80 D1=D1+ 6
```

```
776 1AA90 D0          A=0    A
777 1AA92 14B         A=DAT1 B
778 1AA95 137         CD1EX
779 1AA98 C2          C=C+A  A
780 1AA9A 135         D1=C
781 1AA9D 611F        GOTO   EXPS10
782             *
783 1AAA1             END
```

```
 CHARER  Ext                  -   521
 CHKPSF  Ext                  -    55
=CLLINK  Abs  108332 #1A72C  -   390
 CPL#10  Ext                  -   142
 CURREN  Ext                  -   150   404   608   650
 CURRST  Ext                  -   390
 EXPJ05  Abs  109077 #1AA15  -   722   742
 EXPJ10  Abs  109080 #1AA18  -   724   729   734   739   761
 EXPS10  Abs  108975 #1A9AF  -   685   701   703   713   720   725   781
 EXPS20  Abs  109030 #1A9E6  -   704   689
 EXPS25  Abs  109060 #1AA04  -   716   710   719
 EXPS30  Abs  109086 #1AA1E  -   727   706
 EXPS40  Abs  109116 #1AA3C  -   737   732
 EXPS50  Abs  109181 #1AA7D  -   761   764   770
 EXPS55  Abs  109185 #1AA81  -   763   758
 EXPS60  Abs  109191 #1AA87  -   769   752   754   756
 EXPS70  Abs  109074 #1AA12  -   721   750
 EXPS80  Abs  109197 #1AA8D  -   775   748
=EXPSKP  Abs  108972 #1A9AC  -   684   452   461   490   557   586   736
 FINDAj  Ext                  -   421   746
 FINDL   Ext                  -   109   128
 GETLN#  Abs  108306 #1A712  -   214    98   101   104   116
 ISRAM?  Ext                  -   408
 InvArg  Ext                  -   148
 LASTFN  Ext                  -   743   747
=LINE#1  Abs  108941 #1A98D  -   650    84   135   412   634
 MFERRj  Ext                  -   411
=MFerrj  Abs  108399 #1A76F  -   411
 NXtstm  Ext                  -   211
 PCADDR  Ext                  -    67
 PFNDL*  Ext                  -   635
 POlljj  Ext                  -   518
 PRSCOP  Ext                  -    59
 REN005  Abs  107930 #1A59A  -    63
 REN010  Abs  107969 #1A5C1  -    82    66
 REN020  Abs  108057 #1A619  -   113   111
 REN100  Abs  108110 #1A64E  -   135    99   102   105   117   123   132
 REN105  Abs  108158 #1A67E  -   150   139   147
 REN110  Abs  108185 #1A699  -   158   188
 REN120  Abs  108196 #1A6A4  -   170   203
 REN130  Abs  108229 #1A6C5  -   182
 REN140  Abs  108247 #1A6D7  -   189   181
 REN160  Abs  108265 #1A6E9  -   196   204
 REN170  Abs  108290 #1A702  -   208   172   178
=REN180  Abs  108300 #1A70C  -   211   112
 RENMP   Ext                  -    46
=RENSUB  Abs  108371 #1A753  -   403    65   210
=RENUM   Abs  107910 #1A586  -    47
 RNS150  Abs  108405 #1A775  -   412   409
 RNS160  Abs  108413 #1A77D  -   414   497   614
 RNS170  Abs  108425 #1A789  -   418
 RNS300  Abs  108490 #1A7CA  -   443   429
 RNS310  Abs  108512 #1A7E0  -   452   463
 RNS320  Abs  108519 #1A7E7  -   455   446
 RNS330  Abs  108522 #1A7EA  -   456   453   509
```

RNS340	Abs	108525	#1A7ED	-	457	495					
RNS350	Abs	108529	#1A7F1	-	460	449					
RNS390	Abs	108542	#1A7FE	-	467						
RNS400	Abs	108545	#1A801	-	470	431	433	564			
RNS410	Abs	108579	#1A823	-	486	441	473	482			
RNS500	Abs	108583	#1A827	-	490	435					
RNS510	Abs	108590	#1A82E	-	492	437					
RNS520	Abs	108609	#1A841	-	501	439					
RNS530	Abs	108675	#1A883	-	523	520					
RNS540	Abs	108680	#1A888	-	525	486	506	517			
RNS550	Abs	108684	#1A88C	-	542	512	515				
RNS560	Abs	108720	#1A8B0	-	555	550	561				
RNS570	Abs	108723	#1A8B3	-	557	545	553				
RNS600	Abs	108743	#1A8C7	-	576	423	425	427	457	524	593
RNS610	Abs	108773	#1A8E5	-	586	584					
RNS615	Abs	108777	#1A8E9	-	587	623					
RNS620	Abs	108797	#1A8FD	-	597	525	588				
RNS630	Abs	108824	#1A918	-	607	413					
RNS635	Abs	108848	#1A930	-	614	606					
RNS650	Abs	108852	#1A934	-	618	484	580				
RNS655	Abs	108862	#1A93E	-	622	639	648				
RNS660	Abs	108868	#1A944	-	624	620					
RNS680	Abs	108916	#1A974	-	641	628					
RNUMDC	Ext			-	45						
STMTNF	Ext			-	71						
UPDCRL	Ext			-	70						
a"	Ext			-	687	708					
a'	Ext			-	711						
eFACCS	Ext			-	410						
oFLSTr	Ext			-	656						
oSUBLn	Ext			-	393						
pREN	Ext			-	519						
t%	Ext			-	551						
t@	Ext			-	559						
tADIG9	Ext			-	737						
tARRAY	Ext			-	753						
tCOLON	Ext			-	543						
tCOMMA	Ext			-	216	591					
tDISP	Ext			-	432						
tDMYAR	Ext			-	755						
tELSE	Ext			-	436						
tERROR	Ext			-	444						
tEXTIF	Ext			-	493						
tFN	Ext			-	751						
tGOSUB	Ext			-	422						
tGOTO	Ext			-	424						
tIF	Ext			-	434						
tINTR	Ext			-	507						
tISUB$	Ext			-	757						
tLINE#	Ext			-	480	578					
tLITRL	Ext			-	548	581					
tON	Ext			-	428						
tPRINT	Ext			-	430						
tRELOP	Ext			-	740						
tRESTR	Ext			-	426						

```
tUSING  Ext              -  471
tXFN    Ext              -  749
tXWORD  Ext              -  438
tZ      Ext              -  730
```

Input Parameters

  Source file name is SC&REN::MS

  Listing file name is SC/REN:TI:ML::-1

  Object file name is SC%REN:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                    TITLE  Key Routines <831212.1207>
1AAA1               ABS    #1AAA1
*     SSS    GGG    &     K   K  EEEEE  Y   Y
*    S   S  G   G  & &    K  K   E      Y   Y
*    S      G      & &    K K    E       Y Y
*     SSS   G GGG   &     KK     EEEE     Y
*        S  G   G  & & &  K K    E        Y
*    S   S  G   G  &  &   K  K   E        Y
*     SSS    GGG    && &  K   K  EEEEE    Y

                    RDSYMB TI%EQU::MS
**********************************************************************
**********************************************************************
**
** Name:    FTCHKY  -  Fetch Key
**
** Category:   STEXEC
**
** Purpose:    FETCHES KEY DEFINITION
**
** Entry:    2 ENTRY POINTS:
**             1) FTCHKY - Entry point for FETCH KEY statement
**                         DO MUST POINT PAST tKEY
**                         S7=0
**             2) FTKYOO - Entry point for LIST of KEY files
**                         S7 Must be set & A(B)= Keycode
**                         D1 at start of key assignment entry
**
** Exit:     'DEF KEY <string>,<string>' Output to INBS
**           R0 Points to end of key assignment entry
**
**           S7=1 on entry=> Carry set
**             =0 on entry=> exit via DSPLIN
**
** Calls:    GTKYCD, OUTNBS, OUTBYT, KEYFND, LSTLEN,
**           K#2ASC, OUTC15, D=AVME, LDCSET, KYTYP1
**
** Uses:     A-D, D1,D0, R0-R3, S0
**
** Stack lvls: 6 (LSTLEN uses 5)
**
** Detail:   If enter at FTKYOO with S7 set, won't use R1-R3
**
** History:
**
**    Date     Programmer   Modifications
** --------   ----------   -------------
** 07/06/82   S.W.         Improved documentation
**
**********************************************************************
**********************************************************************
*
*
1AAA1 8F00 =FTCHKY GOSBVL =GTKYCD
      000
```

```
 55 1AAA8 D4          A=B    A
 56 1AAAA 101         R1=A                    SAVE KEY #
 57           * On FETCH KEY ... , know S7 is already clear  (RUN loop)
 58 1AAAD 8F00  FTKY00 GOSBVL =LDCSET         Set D @ AVMEME, DO @ OUTBS
          000
 59 1AAB4 3F44        LCASC  \ YEK FED\
          5464
          02B4
          5495
          02
 60 1AAC6 D8          B=A    A
 61 1AAC8 8F00        GOSBVL =OUTC15
          000
 62 1AACF D4          A=B    A
 63           *
 64 1AAD1 79F0        GOSUB  K#2ASC
 65 1AAD5 31C2        LCASC  \,\
 66 1AAD9 7021        GOSUB  OUtbyt
 67 1AADD 877         ?ST=1  7                CALLED BY LIST?
 68 1AAE0 D2          GOYES  FTKY20
 69           *
 70 1AAE2 132         ADOEX
 71 1AAE5 121         AR1EX                   SAVE DO; RESTORE KEY CODE
 72 1AAE8 D8          B=A    A
 73 1AAEA 8E00        GOSUBL =keyfnd          SEE IF THERE'S AN ASSIGNMENT
          00
 74 1AAF0 111         A=R1
 75 1AAF3 137         CD1EX
 76 1AAF6 1F00        D1=(5) =LDCSPC
          000
 77 1AAFD 141         DAT1=A A                Save cursor position
 78 1AB00 135         D1=C
 79 1AB03 130         DO=A                    RESTORE DO
 80 1AB06 7000        GOSUB  =D=AVME          Restore D(A); doesn't affect carry
 81 1AB0A 5C6         GONC   FTKY-1           CARRY CLR=> NO ASSIGNMENT
 82           * KNOW S8 IS CLEAR AT THIS POINT
 83           * D1 POINTS AT KEY ASSIGNMENT
 84 1AB0D 133  FTKY20 AD1EX
 85 1AB10 131         D1=A
 86 1AB13 100         R0=A                    PTR TO START OF ASSIGNMENT ENTRY
 87 1AB16 171         D1=D1+ 2
 88 1AB19 D2          C=0    A
 89 1AB1B 14F         C=DAT1 B                ENTRY LENGTH
 90 1AB1E CA          A=A+C  A
 91 1AB20 D8          B=A    A                PTR TO END OF KEY ASSIGNMENT ENTRY
 92           * CHECK FOR SINGLE QUOTE IN ASSIGNMENT
 93 1AB22 840         ST=0   0
 94 1AB25 3172        LCASC  \'\
 95 1AB29 170         D1=D1+ 1
 96           *
 97 1AB2C 171  FTKY30 D1=D1+ 2
 98 1AB2F 133         AD1EX
 99 1AB32 8B8         ?A>=B  A                DONE SEARCHING?
100 1AB35 31          GOYES  FTKY50
101 1AB37 131         D1=A
```

```
102 1AB3A 14B          A=DAT1 B
103 1AB3D 966          ?A#C   B
104 1AB40 CE           GOYES  FTKY30
105               *
106 1AB42 850          ST=1   0              SET SINGLE QUOTE FOUND FLAG
107 1AB45 302          LCHEX  2              DELIMITER BECOMES DOUBLE QUOTE
108               *
109 1AB48 71B0  FTKY50 GOSUB  OUtbyt
110 1AB4C D4           A=B    A              END OF ASSIGNMENT
111 1AB4E 120          AROEX                 A(A)=BGN OF ASSIGN.; R0=ASSIGN. END
112 1AB51 131          D1=A
113 1AB54 171          D1=D1+ 2
114 1AB57 14B          A=DAT1 B              ENTRY LENGTH
115 1AB5A 3150         LC(2)  5
116 1AB5E B6E          C=A-C  B              STRING LENGTH
117 1AB61 171          D1=D1+ 2              STEP OVER LENGTH
118 1AB64 1574         C=DAT1 S              READ IN TERMINATOR ENCODING
119 1AB68 AC7          D=C    S
120 1AB6B 170          D1=D1+ 1
121 1AB6E 816          CSRC                  SHIFT #NIBS TO WRITE INTO NIB 14
122 1AB71 816          CSRC                  SHIFT #16-NIB BLKS INTO SIGN FIELD
123 1AB74 5B0          GONC   FTKY60         (B.E.T.)
124               *
125 1AB77 181   FTKY-1 D0=D0- 2
126 1AB7A AC2          C=0    S
127 1AB7D 573          GONC   FTKY81         (B.E.T.)
128               *
129 1AB80 A4E   FTKY60 C=C-1  S
130 1AB83 491          GOC    FTKY70
131 1AB86 1537         A=DAT1 W
132 1AB8A 2F           P=     15
133 1AB8C 7600         GOSUB  outnbs
134 1AB90 17F          D1=D1+ 16
135 1AB93 5CE          GONC   FTKY60         (B.E.T.)
136               *
137 1AB96 8D00  outnbs GOVLNG =OUTNBS
          000
138               *
139 1AB9D 80DE  FTKY70 P=C    14
140 1ABA1 0D           P=P-1
141 1ABA3 4A0          GOC    FTKY80
142 1ABA6 1531         A=DAT1 WP
143 1ABAA 78EF         GOSUB  outnbs
144               *
145 1ABAE 7D30  FTKY80 GOSUB  KY"CK
146 1ABB2 ACB          C=D    S              Given key definition type,
147 1ABB5 8E00  FTKY81 GOSUBL =KYTYP1         KEYTYP returns ascii terminator
          00
148               *
149 1ABBB 8F00         GOSBVL =LSTLN+         OUTPUT TERMINATOR & CALC. BUF LEN
          000
150 1ABC2 877          ?ST=1  7
151 1ABC5 00           RTNYES                LIST?
152 1ABC7 8D00         GOVLNG =DSPLI+
          000
```

```
153             *
154             *
155             ****************************************************************
156             ****************************************************************
157             **
158             ** Name:    K#2ASC  -  Key number to ascii conversion
159             **
160             ** Category:   LOCAL
161             **
162             ** Purpose:
163             **      1) converts keycode to ascii equivalent
164             **      2) outputs correct deliniting quote for ascii key name
165             **
166             ** Entry:
167             **      2 entry points:
168             **        1) K#2ASC - keycode in A(B), P=0
169             **        2) KY"CK  - S0=0 => single quote, else double quote
170             **
171             ** Exit:
172             **      P=0, carry clear
173             **      via OUTBYT
174             **
175             ** Calls:      KEYNAM, OUTNBS, KY"CK
176             **
177             ** Uses:       A, B, C, R0, S0, S1, S2, D0
178             **
179             ** Stk lvls:   4
180             **
181             ** History:
182             **
183             **    Date     Programmer              Modification
184             **  --------   ----------   ----------------------------------
185             **  07/06/82   S.W.         Added documentation
186             **
187             ****************************************************************
188             ****************************************************************
189             *
190 1ABCE 136   K#2ASC CDOEX
191 1ABD1 06          RSTK=C                  SAVE D0
192 1ABD3 7D20        GOSUB  KEYNAM
193 1ABD7 07          C=RSTK
194 1ABD9 134         D0=C                    RESTORE D0
195 1ABDC 80FF        CPEX   15               SAVE P
196 1ABE0 7B00        GOSUB  KY"CK            OUTPUT 1ST DELIMITER
197 1ABE4 AEA         A=C    B
198 1ABE7 80DF        P=C    15               RESTORE P
199 1ABEB 77AF        GOSUB  outnbs
200             *
201 1ABEF 20    KY"CK  P=     0
202 1ABF1 3172        LCASC  \'\
203 1ABF5 860         ?ST=0  0
204 1ABF8 50          GOYES  OUtbyt
205 1ABFA 302         LC(1)  \"\
206             *
207 1ABFD 8D00 =OUtbyt GOVLNG =OUTBYT
```

```
            000
208             *-
209             *-
```

```
                        EJECT
                *****************************************************************
                *****************************************************************
                **
                ** Name:(S) KEYNAM  -  Return key name string from keycode
                **
                ** Category:   EXCUTL
                **
                ** Purpose:
                **      Returns string representing a keycode
                **
                ** Entry:
                **      A(B)=Keycode to be named.
                **
                ** Exit:
                **      A(WP)=ASCII for keycode.
                **      P=Word thru pointer length of text
                **      UseQuo(S0) set iff double quotes should be used
                **      to surround string.
                **
                ** Calls:      RANGE, HXDASC
                **
                ** Uses:       A,B,C,R0,S0,S1,S2,D0
                **
                ** Stk lvls: 2
                **
                ** History:
                **
                **    Date     Programmer              Modification
                ** ---------   ----------   ---------------------------------
                ** 11/10/83    B.S.         Updated documentation
                **
                *****************************************************************
                *****************************************************************
                 UseQuo EQU    0
                 FShift EQU    1
                 GShift EQU    2
1AC04 100       =KEYNAM R0=A
1AC07 D1                B=0    A
1AC09 AE8               B=A    B
1AC0C 840               ST=0   UseQuo
1AC0F 841               ST=0   FShift
1AC12 842               ST=0   GShift
1AC15 3187              LC(2)  123
1AC19 966               ?A#C   B
1AC1C 50                GOYES  KEYN20
1AC1E 850       KEYN10  ST=1   UseQuo
1AC21 3400      KEYN20  LC(5)  =KEYCOD
      000
1AC28 C9                C=C+B  A
1AC2A C9                C=C+B  A                Calculate table address.
1AC2C 134               D0=C
1AC2F 14A               A=DAT0 B                Read table entry
1AC32 3302              LC(4)  \}\~\ \          Range of ASCII characters
      D7
```

```
263 1AC38 7000          GOSUB  =RANGE        Is it in ASCII range?
264 1AC3C 5F1           GONC   KEYN40         Yes, then prepare to return
265 1AC3F 3183  KEYN30  LC(2)  56
266 1AC43 9ED           ?B<=C  B              Is it unshifted?
267 1AC46 63            GOYES  KEYN60         Yes=> nothing to subtract from KC
268 1AC48 851           ST=1   FShift
269 1AC4B B61           B=B-C  B              Subtract off f shift
270 1AC4E 9ED           ?B<=C  B
271 1AC51 0D            GOYES  KEYN20         No, then try to find an ASCII again
272 1AC53 852           ST=1   GShift
273 1AC56 B61           B=B-C  B              Subtract off g shift
274 1AC59 57C           GONC   KEYN20         B.E.T.  Try to find an ASCII again
275             *-
276             *-
277 1AC5C 3166  KEYN40  LCASC  \f\
278 1AC60 21            P=     1
279 1AC62 861           ?ST=0  FShift         Is it unshifted?
280 1AC65 00            RTNYES                Yes, then return
281 1AC67 862           ?ST=0  GShift         Is it a GShift?
282 1AC6A 50            GOYES  KEYN50
283 1AC6C B66           C=C+1  B
284 1AC6F BF0   KEYN50  ASL    W
285 1AC72 BF0           ASL    W
286 1AC75 AEA           A=C    B
287 1AC78 23            P=     3
288 1AC7A 02            RTNSC
289             *-
290             *-
291 1AC7C 110   KEYN60  A=R0                  Recall keycode
292 1AC7F AF1           B=0    W
293 1AC82 8F00          GOSBVL =HXDASC        Convert Hex to Decimal ASCII
          000
294 1AC89 3132          LCASC  \#\
295 1AC8D 2F            P=     15
296 1AC8F B95   KEYN70  BSR    WP
297 1AC92 0D            P=P-1
298 1AC94 B95           BSR    WP
299 1AC97 0D            P=P-1
300 1AC99 929           ?B=0   XS
301 1AC9C 3F            GOYES  KEYN70         Loop til non-leading-zero found
302 1AC9E AE5           B=C    B
303 1ACA1 AF4           A=B    W
304 1ACA4 03            RTNCC
305             *
```

```
306                     EJECT
307                 ****************************************************************
308                 ****************************************************************
309                 **
310                 ** Name:(S) KEY$    -  KEY$ function
311                 **
312                 ** Category:   FNEXEC
313                 **
314                 ** Purpose:
315                 **      Evaluates KEY$ function
316                 **
317                 ** Entry:
318                 **      P      =  0
319                 **
320                 ** Exit:
321                 **      P      =  0
322                 **      via ADHEAD
323                 **
324                 ** Calls:      D=AVMS, POPBUF, KEYNAM, STKCHR
325                 **
326                 ** Uses:       A-C,D(A), R0-R2, S0-S2, D1,D0
327                 **
328                 ** Stk lvls:   3
329                 **
330                 ** History:
331                 **
332                 **    Date     Programmer              Modification
333                 **  --------   ----------   ----------------------------------
334                 **  08/29/83   S.W.         Added documentation header
335                 **
336                 ****************************************************************
337                 ****************************************************************
338 1ACA6 00               NIBHEX 00
339 1ACA8 136   =KEY$      CDOEX
340 1ACAB 10A               R2=C                    Save D0 in R2
341 1ACAE 8E00              GOSUBL =D=AVMS
            00
342 1ACB4 137               CD1EX
343 1ACB7 135               D1=C
344 1ACBA 109               R1=C                    Copy D1 to R1
345 1ACBD 8F00              GOSBVL =POPBUF          Pop key from buffer
            000
346 1ACC4 402               GOC    KEY$10           Skip if buffer empty
347 1ACC7 D4                A=B    A                Copy keycode to A(B)
348 1ACC9 773F              GOSUB  KEYNAM           Get key name
349 1ACCD A96               C=A    WP
350 1ACD0 8E00    KEY$05    GOSUBL =STKCHR          Stack a character
            00
351 1ACD6 BF6               CSR    W
352 1ACD9 BF6               CSR    W                Shift one character
353 1ACDC 0D                P=P-1
354 1ACDE 0D                P=P-1                   Decrement count
355 1ACE0 5FE               GONC   KEY$05           Loop back until all chars out
356 1ACE3 20                P=     0
357 1ACE5 11A   KEY$10      C=R2
```

```
358 1ACE8 134          D0=C                    Restore D0
359 1ACEB 840          ST=0   0                Don't return from ADHEAD
360 1ACEE 8C00 =ADHDj  GOLONG =ADHEAD          Add string header
          00
361            *-
362            *-
```

```
363                     EJECT
364                 ****************************************************************
365                 ****************************************************************
366                 **
367                 ** Name:    LSTKEY  -  List Key
368                 **
369                 ** Category:   STEXEC
370                 **
371                 ** Purpose:    LISTS A KEY FILE
372                 **
373                 ** Entry:      D1 POINTS TO FILE HEADER
374                 **
375                 ** Exit:       Exits via NXTSTM
376                 **
377                 ** Calls:      FTCHKY, PRPSND, KYARGS, CK"ON", PARMCK
378                 **
379                 ** Uses:       A-D, R0-R3, D1, D0, S7, S8, S14, S-R1-1
380                 **
381                 ** Stack lvls: 7
382                 **
383                 ** History:
384                 **
385                 **    Date      Programmer   Modifications
386                 **  --------    ----------   -------------
387                 **  07/06/82    S.W.         Improved documentation
388                 **  10/14/82    S.W.         Interfaced with SENDWD
389                 **  12/17/82    S.W.         Call CK"ON" instead of CKEXCP;
390                 **                            Check carry instead of S14
391                 **
392                 ****************************************************************
393                 ****************************************************************
394                 *
395 1ACF4 7730 =LSTKEY GOSUB  KYARGS
396                 *
397 1ACF8 DB          C=D    A              Pointer to file end
398 1ACFA 1B00        D0=(5) =S-R1-1
            000
399 1AD01 144         DAT0=C A
400                 *
401 1AD04 857   LSTK10 ST=1   7
402 1AD07 72AD        GOSUB  FTKY00
403                 *
404 1AD0B 8F00        GOSBVL =PRPSND
            000
405 1AD12 8F00        GOSBVL =CK"ON"
            000
406 1AD19 511         GONC   LSTKDN          ATTN key down?
407 1AD1C 135         D1=C
408 1AD1F D0          A=0    A
409 1AD21 14B         A=DAT1 B               Want keycode in A(A)
410 1AD24 7C30        GOSUB  PARMCK
411 1AD28 5BD         GONC   LSTK10          (B.E.T.)
412                 *
413                 *
414 1AD2B 6000  LSTKDN GOTO   =REN180        GOTO  NXTSTM
```

```
                *
                *
                ********************************************************************
                ********************************************************************
                **
                ** Name:    KYARGS  -  Interpret Key Number Argument
                **
                ** Category:   FILUTL
                **
                ** Purpose:
                **      Interprets key# range & applies it to specified KEY file
                **
                ** Entry:
                **      R3 contains upper limit of range (decimal)
                **      2 entry points:
                **       1) KYARGS -  R1 contains lower limit of range (dec)
                **                    D1 points at header of key file
                **                    S10=1 => only 1 parm specified
                **       2) PARMCK -  A(A)=next keycode in file to consider
                **
                ** Exit:
                **      P=0
                **      D(A) =  pointer to end of file
                **      R3 contains upper limit of range in hex
                **      Carry set=> match on 1st parm
                **                  D1 at assignment entry
                **                  A(A)=B(A)=corresponding keycode
                **                  C(A)=pointer to end of file
                **            clr=> D1 at next largest entry in range
                **                  A(A)=corresponding keycode
                **                  C(A)=upper limit of range
                **
                **      If match on single parm (S10=1) isn't found, or
                **      if no entry within specified range, exits via NXTSTM
                **
                ** Calls:      DECHEX, KEYFND
                **
                ** Uses:       A(A), B(A), C(A), D(A), D1, D0, R1, R3
                **
                ** Stk lvls:   1 (assuming DECHEX uses 0)
                **
                ** Note:       Assumes PARMXQ was called for R1,R3 set-up
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   ----------------------------------
                **  07/06/82   S.W.         Improved documentation.
                **                           Modified entry conditions to PARMCK
                **                           from A(B)=keycode to A(A)=keycode.
                **
                ********************************************************************
                ********************************************************************
                *
1AD2F AF0   =KYARGS A=0     W             WANT UPPER NIBS ZEROED OUT
```

```
470 1AD32 119          C=R1
471 1AD35 7000         GOSUB  =DCHX=C
472 1AD39 D6           C=A    A
473 1AD3B D7           D=C    A              SAVE KEYCODE
474 1AD3D 11B          C=R3
475 1AD40 7000         GOSUB  =DCHX=C
476 1AD44 103          R3=A
477 1AD47 133          AD1EX                 PTR TO FILE HEADER
478 1AD4A 8F00         GOSBVL =KYFND+
          000
479               * Ptr to file end in D(A)
480               *       C=D    A
481               *       R1=C                  SAVE IT
482 1AD51 400          RTNC                  FOUND MATCH?
483               * MATCH NOT FOUND
484 1AD54 87A          ?ST=1  10             EXACTLY 1 PARM SPECIFIED?
485 1AD57 4D           GOYES  LSTKDN
486               * SEE IF BIGGER ASSIGNED KEY CODE EXISTS
487 1AD59 137          CD1EX
488 1AD5C 135          D1=C
489 1AD5F 8BF          ?C>=D  A              POINTING AT END OF FILE?
490 1AD62 9C           GOYES  LSTKDN
491               * BIGGER KEYCODE POINTED TO - SEE IF IT'S WITHIN RANGE
492 1AD64 11B   PARMCK C=R3
493 1AD67 8B6          ?A>C   A
494 1AD6A 1C           GOYES  LSTKDN
495 1AD6C 03           RTNCC
496               *
```

```
=ADHDj   Abs  109806 #1ACEE -   360
 ADHEAD  Ext                -   360
 CK"ON"  Ext                -   405
 D=AVME  Ext                -    80
 D=AVMS  Ext                -   341
 DCHX=C  Ext                -   471   475
 DSPLI+  Ext                -   152
 FShift  Abs       1 #00001 -   245   251   268   279
=FTCHKY  Abs  109217 #1AAA1 -    54
 FTKY-1  Abs  109431 #1AB77 -   125    81
 FTKY00  Abs  109229 #1AAAD -    58   402
 FTKY20  Abs  109325 #1AB0D -    84    68
 FTKY30  Abs  109356 #1AB2C -    97   104
 FTKY50  Abs  109384 #1AB48 -   109   100
 FTKY60  Abs  109440 #1AB80 -   129   123   135
 FTKY70  Abs  109469 #1AB9D -   139   130
 FTKY80  Abs  109486 #1ABAE -   145   141
 FTKY81  Abs  109493 #1ABB5 -   147   127
 GShift  Abs       2 #00002 -   246   252   272   281
 GTKYCD  Ext                -    54
 HXDASC  Ext                -   293
 K#2ASC  Abs  109518 #1ABCE -   190    64
=KEY$    Abs  109736 #1ACA8 -   339
 KEY$05  Abs  109776 #1ACD0 -   350   355
 KEY$10  Abs  109797 #1ACE5 -   357   346
 KEYCOD  Ext                -   257
 KEYN10  Abs  109598 #1AC1E -   256
 KEYN20  Abs  109601 #1AC21 -   257   255   271   274
 KEYN30  Abs  109631 #1AC3F -   265
 KEYN40  Abs  109660 #1AC5C -   277   264
 KEYN50  Abs  109679 #1AC6F -   284   282
 KEYN60  Abs  109692 #1AC7C -   291   267
 KEYN70  Abs  109711 #1AC8F -   296   301
=KEYNAM  Abs  109572 #1AC04 -   247   192   348
 KY"CK   Abs  109551 #1ABEF -   201   145   196
=KYARGS  Abs  109871 #1AD2F -   469   395
 KYFND+  Ext                -   478
 KYTYP1  Ext                -   147
 LDCSET  Ext                -    58
 LDCSPC  Ext                -    76
 LSTK10  Abs  109828 #1AD04 -   401   411
 LSTKDN  Abs  109867 #1AD2B -   414   406   485   490   494
=LSTKEY  Abs  109812 #1ACF4 -   395
 LSTLN+  Ext                -   149
 OUTBYT  Ext                -   207
 OUTC15  Ext                -    61
 OUTNBS  Ext                -   137
=OUtbyt  Abs  109565 #1ABFD -   207    66   109   204
 PARMCK  Abs  109924 #1AD64 -   492   410
 POPBUF  Ext                -   345
 PRPSND  Ext                -   404
 RANGE   Ext                -   263
 REN180  Ext                -   414
 S-R1-1  Ext                -   398
 STKCHR  Ext                -   350
```

UseQuo	Abs	0	#00000	-	244	250	256	
keyfnd	Ext			-	73			
outnbs	Abs	109462	#1AB96	-	137	133	143	199

Input Parameters

  Source file name is SG&KEY::MS

  Listing file name is SG/KEY:TI:ML::-1

  Object file name is SG%KEY:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                      TITLE  VAL Function <831212.1206>
 2 1AD6E                ABS    #1AD6E
 3              *     SSS   BBBB    &     V   V    A    L
 4              *    S   S  B   B  & &    V   V   A A   L
 5              *    S      B   B  & &    V   V  A   A  L
 6              *     SSS   BBBB    &      V V   A   A  L
 7              *        S  B   B  & & &   V V   AAAAA  L
 8              *    S   S  B   B  &  &     V    A   A  L
 9              *     SSS   BBBB    && &    V    A   A  LLLLL
10
11                      RDSYMB SB%RAM::MS
12             =ValSub EQU    10              Indicates VAL is used as a sub
13             *
14             *
15 1AD6E 86A   VALERR ?ST=0  ValSub
16 1AD71 80            GOYES  invarg
17 1AD73 8C00          GOLONG =RDATTY
         00
18 1AD79 8C00  invarg  GOLONG =InvArg          Invalid argument error
         00
19             *-
20             *-
21 1AD7F 411           NIBHEX 411
22 1AD82 84A   =VAL    ST=0   ValSub           VAL is not being used as a sub
23 1AD85 7600          GOSUB  VAL00            VAL statement execution
24 1AD89 8C00  EXPRj   GOLONG =expr
         00
25             *-
26             *-
27             ************************************************************
28             ************************************************************
29             **
30             ** Name:(S) VAL00   -  Parse and Execute a String on Stack
31             **
32             ** Category:   EXCUTL
33             **
34             ** Purpose:
35             **      System VAL function.  Converts a string into a number.
36             **      Any valid numeric expression may be passed.
37             **
38             ** Entry:
39             **      P      =  0
40             **      D1 points to string on top of math stack.
41             **      ST10 (=ValSub) set iff VAL is being called
42             **         as a subroutine.
43             **         Will cause "Data Type" error instead of
44             **         "Invalid Argument" and will require the
45             **         valid expression to be followed by a CR.
46             **
47             ** Exit:
48             **      P      =  0
49             **      String on top of stack has been replaced by the value
50             **      obtained by parsing and executing the string.
51             **
52             **
```

```
                ** Calls:      XXHEAD,STKCHR,ADHEAD,REVPOP,EXCPAR,OUTBYT,
                **             MOVED2,PSHSTK,EXPR,POPSTK,POP1N,AVE=D1,MFERR
                **
                ** Uses.......
                **  Inclusive: A,B,C,D,R0,R1,R2,R3,R4,D1
                **
                ** Stk lvls:   4
                **
                ** NOTE:
                **      This routine calls expression execute which may call a
                **      user defined function; this may alter a lot of RAM
                **      locations.  The D0 that is passed in is kept on the
                **      GOSUB stack so it will be updated if memory moves.
                **
                ** Algorithm:
                **      Appends a CR to string on stack.
                **      Reverses string.
                **      Parses string and verifies it is a valid numeric expr.
                **      Appends an EOL to parsed code.
                **      Moves parsed code onto stack, covering original string.
                **      Saves 2 RSTK levels and D0 (PC) on GOSUB stack.
                **      Calls EXPR to evaluate expression.
                **      Pops value from stack.
                **      Collapses parsed code from stack.
                **      Checks validity of pointers saved on GOSUB stack
                **        and jumps to MFERR(eMMCOR) if any are not valid.
                **      Restores 2 RSTK levels and D0 (PC) from GOSUB stack.
                **      Pushes value on stack.
                **      Returns
                **
                ** History:
                **
                **    Date      Programmer          Modification
                **  --------    ----------  ----------------------------------
                **  02/04/83    B.S.        Added documentation
                **  04/08/83    B.S.        Modified routine to observe S10.
                **
                ********************************************************************
                ********************************************************************
                 Return EQU    0
 1AD8F 850      =VAL00 ST=1    Return
 1AD92 8E00             GOSUBL =XXHEAD       Remove string header,
       00
                                               set up for STKCHR
 1AD98 31D0             LCHEX  0D            Carriage return
 1AD9C 8E00             GOSUBL =STKCHR       Append CR to end of string
       00
 1ADA2 7000             GOSUB  =ADHDj        Put on new string header
 1ADA6 8F00             GOSBVL =REVPOP       Reverse string, pop it, C=D0
       000
 1ADAD 1B0C             D0=(5) =FUNCD1
       8F2
 1ADB4 144              DAT0=C A             Save old D0 in FUNCD1
 1ADB7 102              R2=A                 Save string length in R2
 1ADBA 8E00             GOSUBL =r<rstk       Save an RSTK level
```

```
                00
    104 1ADC0 8E00          GOSUBL =EXCPAR          Call Execution EXPPAR
                00
    105 1ADC6 86A           ?ST=0  ValSub           VAL used as a sub?
    106 1ADC9 B0            GOYES  VAL01            No, then skip
    107 1ADCB 3100          LC(2)  =tEOL
    108 1ADCF 966           ?A#C   B                Is expression followed by CR?
    109 1ADD2 C9            GOYES  VALERR           No, then error
    110 1ADD4 870   VAL01   ?ST=1  0                Valid expression?
    111 1ADD7 79            GOYES  VALERR           No, then error
    112 1ADD9 863           ?ST=0  3                Numeric expression?
    113 1ADDC 29            GOYES  VALERR           No, then error
    114 1ADDE 3100          LC(2)  =tEOL
    115 1ADE2 7000          GOSUB  =OUtbyt          Append EOL token to expression
    116 1ADE6 DB            C=D    A                C(A)=(MTHSTK)
    117 1ADE8 112           A=R2                    Recall string length
    118 1ADEB C2            C=C+A  A                Add string length
    119 1ADED 135           D1=C                    D1=(Stack pointer)
    120             * Prepare to move code buffer onto stack
    121 1ADF0 132           ADOEX                   A(A)=End of Source
    122 1ADF3 11B           C=R3                    C(A)=Start of Source
    123 1ADF6 7000          GOSUB  =MOVED2          Move output buffer to stack
    124 1ADFA 137           CD1EX                   C(A)=Start of dest
    125 1ADFD 1B99          DO=(5) =MTHSTK
                5F2
    126 1AE04 144           DAT0=C A                Write out pointer to code buffer
    127 1AE07 8E00          GOSUBL =rstk<r
                00
    128             * Save two RSTK levels and old DO(program PC) on GOSUB stack
    129 1AE0D D2            C=0    A
    130 1AE0F 3121          LC(2)  18               18 nibbles
    131 1AE13 D5            B=C    A                B(A)=Amount mem needs to open up
    132 1AE15 22            P=     ((GSBSTK)-(MTHSTK))/5  Adjust 3 pointers
    133                                               (AVMEME,TFORN,GSBSTK)
    134 1AE17 1A3A          DO=(4) =GSBSTK
                5F
    135                                             DO points at last ptr (GSBSTK)
    136 1AE1D 8F00          GOSBVL =PSHSTK
                000
    137 1AE24 AF2           C=0    W
    138 1AE27 A7E           C=C-1  W
    139 1AE2A 1557          DAT1=C W                Write out 3 "F" GOSUB stack
    140                                                entry types
    141 1AE2E 170           D1=D1+ 1                Skip first entry type
    142 1AE31 07            C=RSTK
    143 1AE33 145           DAT1=C A                Save an RSTK level
    144 1AE36 175           D1=D1+ 6                Point to next entry
    145 1AE39 07            C=RSTK
    146 1AE3B 145           DAT1=C A                Save another RSTK level
    147 1AE3E 175           D1=D1+ 6                Point to next entry
    148 1AE41 1B0C          DO=(5) =FUNCD1
                8F2
    149 1AE48 146           C=DAT0 A                Recall old DO(program PC)
    150 1AE4B 145           DAT1=C A                Save old DO(program PC)
    151 1AE4E 1A99          DO=(4) =MTHSTK
```

```
            5F
    152 1AE54 146           C=DAT0 A
    153 1AE57 134           DO=C                    Set PC at start of code buffer
    154 1AE5A 135           D1=C                    Set Stack pointer to end AVMEM
    155 1AE5D 782F          GOSUB  EXPRj            Evaluate expression
    156               *
    157               * VALCHK can not be moved since it is checked in AB%REG to
    158               * determine if STMTDO needs to be updated by during
    159               * user-defined function execution.
    160               *
    161 1AE61 163   =VALCHK DO=DO+ 4                EOL+18 nibbles on GOSUB stack
    162                                               take up 20 of nibbles, now move
    163                                               ahead 4 more to point where
    164                                               stack item will go
    165 1AE64 136           CDOEX
    166 1AE67 10A           R2=C                    Save address where stack item
    167                                               will go
    168 1AE6A 1B3A          DO=(5) =GSBSTK
            5F2
    169 1AE71 142           A=DAT0 A
    170 1AE74 130           DO=A
    171 1AE77 166           DO=DO+ 7                Skip to second entry
    172 1AE7A 7470          GOSUB  GETADR
    173 1AE7E 06            RSTK=C                  Restore an RSTK level
    174 1AE80 185           DO=DO- 6                Move back to first entry
    175 1AE83 7B60          GOSUB  GETADR
    176 1AE87 06            RSTK=C                  Restore another RSTK level
    177 1AE89 16B           DO=DO+ 12               Now move to third entry
    178 1AE8C 7260          GOSUB  GETADR
    179 1AE90 DA            A=C    A
    180 1AE92 103           R3=A                    Save old DO(program PC) in R3
    181 1AE95 164           DO=DO+ 5
    182 1AE98 132           ADOEX                   A(A)=End of save space
    183 1AE9B D2            C=0    A
    184 1AE9D 3121          LC(2)  18               18 nibbles
    185 1AEA1 EE            C=A-C  A                C(A)=Start of save space
    186 1AEA3 22            P=     ((GSBSTK)-(MTHSTK))/5  Adjust 3 pointers
    187                                                     (MTHSTK,TFORN,GSBSTK)
    188 1AEA5 8F00          GOSBVL =POPSTK          Discard save space on GOSUB stack
            000
    189 1AEAC 18E           DO=DO- (GSBSTK)-(MTHSTK)+5     Point at MTHSTK
    190 1AEAF 142           A=DAT0 A                Read (MTHSTK)
    191 1AEB2 131           D1=A
    192 1AEB5 8F00          GOSBVL =POP1N           Pop a number off stack
            000
    193 1AEBC 04            SETHEX
    194 1AEBE 11B           C=R3
    195 1AEC1 134           DO=C                    Restore old DO(program PC)
    196 1AEC4 11A           C=R2                    Recall stack pointer
    197 1AEC7 135           D1=C
    198 1AECA 1517          DAT1=A W                Write out real part
    199 1AECE 591           GONC   VALRTN           Return if not complex
    200 1AED1 118           C=R0                    Recall imaginary part
    201 1AED4 1CF           D1=D1- 16
    202 1AED7 1557          DAT1=C W
```

```
    203 1AEDB 1C1          D1=D1- 2
    204 1AEDE 31E0         LCHEX  0E            Complex "tag"
    205 1AEE2 14D          DAT1=C B
    206 1AEE5 B8A          C=-C   P             Return with carry set for complex
    207 1AEE8 1537  VALRTN A=DAT1 W
    208 1AEEC 8C00         GOLONG =AVE=D1       Set MTHSTK back to D1
              00
    209            *-
    210            *-
    211 1AEF2 146   GETADR C=DAT0 A
    212 1AEF5 8AE          ?C#0   A
    213 1AEF8 00           RTNYES
    214 1AEFA 8D00         GOVLNG =CORUPT
              000
```

ADHDJ	Ext			-	98					
AVE=D1	Ext			-	208					
CORUPT	Ext			-	214					
EXCPAR	Ext			-	104					
EXPRJ	Abs	109961	#1AD89	-	24	155				
FUNCD1	Abs	194752	#2F8C0	-	11	100	148			
GETADR	Abs	110322	#1AEF2	-	211	172	175	178		
GSBSTK	Abs	193955	#2F5A3	-	11	132	134	168	186	189
InvArg	Ext			-	18					
MOVED2	Ext			-	123					
MTHSTK	Abs	193945	#2F599	-	11	125	132	151	186	189
OUtbyt	Ext			-	115					
POP1N	Ext			-	192					
POPSTK	Ext			-	188					
PSHSTK	Ext			-	136					
RDATTY	Ext			-	17					
REVPOP	Ext			-	99					
Return	Abs	0	#00000	-	92	93				
STKCHR	Ext			-	97					
=VAL	Abs	109954	#1AD82	-	22					
=VAL00	Abs	109967	#1AD8F	-	93	23				
VAL01	Abs	110036	#1ADD4	-	110	106				
=VALCHK	Abs	110177	#1AE61	-	161					
VALERR	Abs	109934	#1AD6E	-	15	109	111	113		
VALRTN	Abs	110312	#1AEE8	-	207	199				
=ValSub	Abs	10	#0000A	-	12	15	22	105		
XXHEAD	Ext			-	94					
expr	Ext			-	24					
invarg	Abs	109945	#1AD79	-	18	16				
r<rstk	Ext			-	103					
rstk<r	Ext			-	127					
tEOL	Ext			-	107	114				

Input Parameters

  Source file name is SB&VAL::MS

  Listing file name is SB/VAL:TI:ML::-1

  Object file name is SB%VAL:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    M   M  N   N  &      GGG    SSS   BBBB
 2            *    MM MM  N   N  & &    G   G  S   S  B   B
 3            *    M M M  NN  N  & &    G      S      B   B
 4            *    M M M  N N N   &     G GGG   SSS   BBBB
 5            *    M   M  N  NN  & & &  G   G      S  B   B
 6            *    M   M  N   N  & &    G   G  S   S  B   B
 7            *    M   M  N   N   && &   GGG    SSS   BBBB
 8            *
 9                    TITLE  External GOSUB Module<831216.1453>
10 1AF01             ABS    #1AF01
11                   RDSYMB SB%RAM::MS
12           ********************************************************************
13           ********************************************************************
14           **
15           ** Name:(S) MGOSUB  -  Execute A GOSUB From Movable Code
16           **
17           ** Category:   EXCUTL
18           **
19           ** Purpose:
20           **      Allows code which may move (such as code within a
21           **      LEXfile in RAM) to GOSUB to a utility which may
22           **      move it (such as a file expand utility).  The utility
23           **      will return to the LEXfile properly even if it moved.
24           **
25           ** Entry:
26           **        Instead of GOSBVL <address of desired routine>, call
27           **        the routine as follows:
28           **
29           **          GOSBVL =MGOSUB
30           **          CON(5) <address of desired subroutine>
31           **            .
32           **            .
33           **            .
34           **
35           **      All registers and modes should be set up as required by
36           **        the subroutine.
37           **
38           ** Exit:
39           **      Execution resumes at the location following the CON(5)
40           **        at the call to MGOSUB.
41           **      All registers, modes and carry are as returned by
42           **        the subroutine.
43           **
44           ** Calls:      STRALL, PSHMCR, POPGSB, RCLALL (falls through)
45           **
46           ** Uses.......
47           **             RAM: SCRTCH, SCREX0, SCREX1, SCREX2
48           **             Register usage is dictated completely by the
49           **               requested subroutine.
50           **
51           ** Stk lvls:   MAX ( 3, <#levels used by requested subroutine>)
52           **
53           ** NOTE:
54           **      The scratch RAM is used before and after this code
55           **      calls the requested subroutine, but not during.
```

```
                **      Thus the subroutine can use the scratch RAM locally,
                **      but not to pass information back to the calling
                **      routine.  The calling routine obviously cannot keep
                **      anything there which is expected to survive =MGOSUB.
                **
                **      =MGOSUB acts transparently for everything, including
                **      CARRY and SB.
                **
                **      Because the return address is kept in RAM, the called
                **      subroutine will see the return address of MGOSUB, not
                **      of the calling code.  So MGOSUB cannot be used to call
                **      a subroutine which uses the return address as a pointer
                **      to data (such as FINDA, TBLJMP, CALBIN and FPOLL).
                **      Neither POLL nor FPOLL can be called through MGOSUB.
                **
                **      Callers to POLL can breathe easily despite this caveat.
                **      POLL also updates the calling address, and so can be
                **      called directly from movable code.  This is not the
                **      case for FPOLL.
                **
                ** Detail:
                **      Calling sequence:
                **
                **        GOSBVL =MGOSUB
                **        CON(5) <address of desired subroutine>
                **        <execution resumes here after return>
                **
                ** Algorithm:
                **      Stores the return address (address past the CON(5)) on
                **        Gosub stack.
                **      Executes subroutine; address on Gosub stack will be
                **        adjusted as necessary if subroutine does a RFADJ.
                **      Retrieves return address from Gosub stack.
                **      Returns to code which called us.
                **
                ** History:
                **
                **    Date     Programmer            Modification
                **  --------   ----------   ----------------------------------
                **  08/31/82   NM           Wrote
                **
                ********************************************************************
                ********************************************************************
 1AF01 7850 =MGOSUB GOSUB  STRALL          Save everything
 1AF05 07           C=RSTK                 Return address from call to us
 1AF07 134          D0=C
 1AF0A 164          D0=D0+ 5               Spot to return to
 1AF0D 132          ADOEX
 1AF10 20           P=     0
 1AF12 8F00         GOSBVL =PSHMCR         Push on GOSUB stack
       000
 1AF19 BF6          CSR    W               Address we just pushed
 1AF1C 1B23         D0=(5) =RTNHER
       FA1
 1AF23 136          CDOEX                  Hold pushed addr in D0
```

```
 109 1AF26 06           RSTK=C              Return to RTNHER from subroutine
 110 1AF28 184          D0=D0- 5
 111 1AF2B 146          C=DAT0 A            Address of desired subroutine
 112 1AF2E 6220         GOTO   RTNH10       Restore everything & GO
 113 1AF32 7720 =RTNHER GOSUB  STRALL       Save everything
 114 1AF36 8F00  POPADR GOSBVL =POPGSB      Pop address off of stack.
           000
 115 1AF3D 491          GOC    POPERR       Go if stack empty.
 116 1AF40 2F           P=     15
 117 1AF42 308          LCHEX  8
 118 1AF45 947          ?C#D   S            Is this Microcode return?
 119 1AF48 EE           GOYES  POPADR       No.  Pop next thing.
 120 1AF4A DB           C=D    A            Yes.  Fetch address.
 121 1AF4C 8AA          ?C=0   A            Zeroed out?
 122 1AF4F 80           GOYES  POPERR       Yes.  Error.
 123 1AF51 06    RTNH10 RSTK=C              Desired address
 124 1AF53 6B60         GOTO   RCLALL       Restore everything and RTN
 125 1AF57      =MMCORj
 126 1AF57 8C00  POPERR GOLONG =SYSERR
           00
 127           ********************************************************************
 128           ********************************************************************
 129           **
 130           ** Name:    STRALL  -  Save A-D,D0,D1,mode,P,Cry,SB In SCRTCH
 131           **
 132           ** Category:   LOCAL
 133           **
 134           ** Purpose:
 135           **      Store CPU regs, flags and mode in scratch RAM for
 136           **      safekeeping.
 137           **
 138           ** Entry:
 139           **      None.
 140           **
 141           ** Exit:
 142           **      HEX mode.
 143           **
 144           ** Calls:      None.
 145           **
 146           ** Uses.......
 147           **             A,C,D1
 148           **
 149           ** Stk lvls:   1
 150           **
 151           ** NOTE:
 152           **      This is the complement of RCLALL.
 153           **
 154           ** Detail:
 155           **      RAM use:
 156           **        SCRTCH:    A.
 157           **        SCRTCH+16: B.
 158           **        SCRTCH+32: C.
 159           **        SCRTCH+48: D.
 160           **        SCRTCH+64: D1.
 161           **        SCRTCH+80: nib 0 = P
```

```
162                **                  nib 1 = 9 if DEC, F if HEX
163                **                  nib 2 = 0 if carry clear, 9 or F if set
164                **                    (depending on mode on entry).
165                **                  nib 3 = 0 if SB clear, F if set.
166                **        SCRTCH+96: D0.
167                **
168                ** History:
169                **
170                **    Date      Programmer              Modification
171                **  --------   ----------   ----------------------------------
172                **  09/01/82   NM           Wrote.
173                **
174                ********************************************************************
175                ********************************************************************
176 1AF5D 06     STRALL RSTK=C                Hold C[A]
177 1AF5F 137          CD1EX
178 1AF62 1F14         D1=(5) (=SCRTCH)+64
           9F2
179 1AF69 145          DAT1=C A               Store D1
180 1AF6C D2           C=0    A
181 1AF6E 550          GONC   STRA10          C[XS]=0 if carry clear
182 1AF71 A2E          C=C-1  XS              =9 or F if carry set
183 1AF74 A6E    STRA10 C=C-1  B               C[1]=9 if DEC, F if HEX
184 1AF77 80F0         CPEX   0               C[0]=P
185 1AF7B 04           SETHEX
186 1AF7D 23           P=     3
187 1AF7F 832          ?SB=0
188 1AF82 50           GOYES  STRA30          C[3]=0 if SB clear
189 1AF84 A0E          C=C-1  P               =F if SB set
190 1AF87 17F    STRA30 D1=D1+ 16
191 1AF8A 145          DAT1=C A               Store P, mode & flags
192 1AF8D 17F          D1=D1+ 16
193 1AF90 136          CD0EX
194 1AF93 145          DAT1=C A               Store D0
195 1AF96 07           C=RSTK                 Restore C
196 1AF98 1E10         D1=(4) =SCRTCH
           9F
197 1AF9E 1517         DAT1=A W               Store A
198 1AFA2 17F          D1=D1+ 16
199 1AFA5 AF4          A=B    W
200 1AFA8 1517         DAT1=A W               Store B
201 1AFAC 17F          D1=D1+ 16
202 1AFAF 1557         DAT1=C W               Store C
203 1AFB3 17F          D1=D1+ 16
204 1AFB6 AFB          C=D    W
205 1AFB9 1557         DAT1=C W               Store D
206 1AFBD 01           RTN
207                ********************************************************************
208                ********************************************************************
209                **
210                ** Name:    RCLALL  -  Recall CPU State As Saved By STRALL
211                **
212                ** Category:   LOCAL
213                **
214                ** Purpose:
```

```
215                **      Restore A,B,C,D,D0,D1,P, mode, Cry, SB and XM as saved
216                **      in scratch RAM by STRALL.
217                **
218                ** Entry:
219                **      None.
220                **
221                ** Exit:
222                **      Registers and mode restored.
223                **
224                ** Calls:      None.
225                **
226                ** Uses.......
227                **             A,B,C,D,D0,D1,P,mode,SB,XM.
228                **
229                ** Stk lvls:   0
230                **
231                ** NOTE:
232                **      This is the complement of STRALL.
233                **
234                ** History:
235                **
236                **    Date      Programmer              Modification
237                **  --------   ----------   ----------------------------------
238                **  09/01/82   NM           Wrote.
239                **
240                ********************************************************************
241                ********************************************************************
242 1AFBF 1B14  RCLALL D0=(5) (=SCRTCH)+64
          9F2
243 1AFC6 146          C=DAT0 A
244 1AFC9 135          D1=C                    Restore D1
245 1AFCC 18F          D0=D0- 16
246 1AFCF 1567         C=DAT0 W
247 1AFD3 AF7          D=C    W                Restore D
248 1AFD6 18F          D0=D0- 16
249 1AFD9 1567         C=DAT0 W                Restore C
250 1AFDD 18F          D0=D0- 16
251 1AFE0 1527         A=DAT0 W
252 1AFE4 AF8          B=A    W                Restore B
253 1AFE7 18F          D0=D0- 16
254 1AFEA 1527         A=DAT0 W                Restore A
255 1AFEE 06           RSTK=C                  Hold C[A]
256 1AFF0 1A15         D0=(4) (=SCRTCH)+80
          9F
257 1AFF6 146          C=DAT0 A                Read flags, mode and P
258 1AFF9 16F          D0=D0+ 16               Point at D0
259 1AFFC 822          SB=0
260 1AFFF 23           P=     3
261 1B001 B86          CSR    P                Set SB if C[3]#0
262 1B004 21           P=     1
263 1B006 B06          C=C+1  P                Set carry if HEX mode stored
264 1B009 440          GOC    RCLA10
265 1B00C 05           SETDEC
266 1B00E 80D0  RCLA10 P=C    0                Restore P
267 1B012 B26          C=C+1  XS               Restore Carry
```

```
268 1B015 146          C=DAT0 A
269 1B018 134          D0=C                Restore D0
270 1B01B 07           C=RSTK              Restore C
271 1B01D 01           RTN
272 1B01F              END
```

```
=MGOSUB  Abs  110337 #1AF01 -   99
=MMCORj  Abs  110423 #1AF57 -  125
 POPADR  Abs  110390 #1AF36 -  114  119
 POPERR  Abs  110423 #1AF57 -  126  115  122
 POPGSB  Ext                -  114
 PSHMCR  Ext                -  105
 RCLA10  Abs  110606 #1B00E -  266  264
 RCLALL  Abs  110527 #1AFBF -  242  124
 RTNH10  Abs  110417 #1AF51 -  123  112
=RTNHER  Abs  110386 #1AF32 -  113  107
 SCRTCH  Abs  194817 #2F901 -   11  178  196  242  256
 STRA10  Abs  110452 #1AF74 -  183  181
 STRA30  Abs  110471 #1AF87 -  190  188
 STRALL  Abs  110429 #1AF5D -  176   99  113
 SYSERR  Ext                -  126
```

Input Parameters

  Source file name is MN&GSB::MS

  Listing file name is MN/GSB:TI:ML::-1

  Object file name is MNXGSB:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                *     AAA   BBBB    &     U   U  TTTTT  L
                *    A   A  B   B  & &    U   U    T    L
                *    A   A  B   B  & &    U   U    T    L
                *    AAAAA  BBBB    &     U   U    T    L
                *    A   A  B   B  & & &  U   U    T    L
                *    A   A  B   B  &  &   U   U    T    L
                *    A   A  BBBB    && &   UUU     T    LLLLL

                      TITLE  Miscellaneous Utility Routines <831212.1515>
1B01F                 ABS    #1B01F
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) RND-12  -  Round A 12-digit Fp Number
                **
                ** Category:   MTHUTL
                **
                ** Purpose:
                **      Round of a floating-point number at specified digit.
                **
                ** Entry:
                **      A = number (12-digit floating-point).
                **      P points to digit where rounding is to take place.  See
                **        detail, below.
                **
                ** Exit:
                **      P=0.
                **      A=Rounded (not IEEE-rounded) 12-digit form.
                **      If P=15 on entry, no rounding was done.
                **      Carry set iff rounding overflowed (returns MAXREAL).
                **
                ** Calls:      None.
                **
                ** Uses.......
                **              A,B,P.
                **
                ** Stk lvls:   0
                **
                ** Detail:
                **      Typically called after IF12A, which sets P to point at
                **      the first fractional digit.
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ----------------------------------
                **              SA           Wrote
                **  10/17/83    NM           Attempted to document
                **
                ********************************************************************
                ********************************************************************
1B01F 89F       =RND-12 ?P=     15
1B022 64                GOYES   AR25
1B024 890               ?P=     0
1B027 70                GOYES   AR00
```

```
 56 1B029 881          ?P#    1
 57 1B02C 40           GOYES  AR05
 58 1B02E 22    AR00   P=     2
 59 1B030 AF1   AR05   B=0    W              MOVE MANTISSA TO B
 60 1B033 ADC          ABEX   M
 61 1B036 A15   AR10   B=B+B  WP             ROUND DOWN?
 62 1B039 A91          B=0    WP
 63 1B03C 512          GONC   AR20           IF SO, GOTO AR20
 64 1B03F A1D          B=B-1  WP             ROUND UP
 65 1B042 2E           P=     14
 66 1B044 B15          B=B+1  WP             DECADE ROLLOVER?
 67 1B047 561          GONC   AR20           IF NOT, GOTO AR20
 68 1B04A B34          A=A+1  X              INCREMENT EXPONENT
 69 1B04D AB8          B=A    X
 70 1B050 A35          B=B+B  X              POSSIBLE EXPONENT OVERFLOW?
 71 1B053 570          GONC   AR17           IF NOT, GOTO AR17
 72 1B056 939          ?B=0   X
 73 1B059 31           GOYES  AR30
 74 1B05B B05   AR17   B=B+1  P              CREATE UNIT MANTISSA
 75 1B05E AD4   AR20   A=B    M              REPLACE MANTISSA
 76 1B061 95C          ?A#0   M
 77 1B064 40           GOYES  AR25
 78 1B066 D0           A=0    A
 79 1B068 20    AR25   P=     0
 80 1B06A 03           RTNCC                 RETURN CARRY CLEAR
 81 1B06C A1D   AR30   B=B-1  WP             CREATE S9.99999999999E499
 82 1B06F B1C          A=B-A  WP
 83 1B072 20           P=     0
 84 1B074 02           RTNSC                 RETURN CARRY SET
 85            ******************************************************************
 86            ******************************************************************
 87            **
 88            ** Name:(S) RANGE   -  Verify A Byte Is In Certain Range
 89            ** Name:(S) DRANGE  -  Verify A Byte Is In Range "0"-"9"
 90            **
 91            ** Category:   GENUTL
 92            **
 93            ** Purpose:
 94            **      Determine if a byte is in a specified range.
 95            **      Caller supplies range for RANGE.
 96            **      This code supplies range of "0" to "9" for DRANGE.
 97            **
 98            ** Entry:
 99            **      P=0.
100            **      A[B] = byte to be checked.
101            **   RANGE: C[B] = lower bound of range to check,
102            **          C[3-2] = upper bound of range to check.
103            **
104            ** Exit:
105            **      P=0.
106            **      Carry clear if byte in range.
107            **
108            ** Calls:      None.
109            **
110            ** Uses.......
```

```
                **              C[A].
                **
                ** Stk lvls:    0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ----------------------------------
                **              SA            Wrote
                **  10/17/83    NM            Attempted to document
                **
                ********************************************************************
                ********************************************************************
1B076 3303 =DRANGE LCASC  \90\            CHECK FOR ASCII DIGIT
        93
1B07C 9E2  =RANGE  ?A<C   B               CHECK FOR BYTE IN RANGE
1B07F 00           RTNYES                 IF NOT, CARRY SET ON RETURN
1B081 F6           CSR    A
1B083 BB6          CSR    X
1B086 B62          C=C-A  B
1B089 01           RTN
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) MEMBER  -  Check If Byte Is A Member Of A Set
                **
                ** Category:   GENUTL
                **
                ** Purpose:
                **      Determine if a byte is a member of a set of bytes.
                **
                ** Entry:
                **      C=set of bytes (C[1-0], C[3-2], etc.).
                **      P points to hinibble of upper byte of set.
                **      A[B] = byte to be checked.
                **
                ** Exit:
                **      P=0.
                **      Carry clear if byte in set.
                **
                ** Calls:      None.
                **
                ** Uses.......
                **              C[WP] (whatever P was on entry), P.
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ----------------------------------
                **              SA            Wrote
                **  10/17/83    NM            Attempted to document
                **
                ********************************************************************
```

```
165                ****************************************************************
166 1B08B B96   MBR10  CSR    WP              SHIFT IN NEXT ELEMENT
167 1B08E 0D           P=P-1
168 1B090 B96          CSR    WP
169 1B093 0D           P=P-1
170 1B095 470          GOC    MBR20
171 1B098 966  =MEMBER ?A#C   B               SET ELEMENT IDENTIFIED?
172 1B09B 0F           GOYES  MBR10           IF NOT, REPEAT MBR10
173 1B09D 20    MBR20  P=     0
174 1B09F 01           RTN                    RETURN
175                ****************************************************************
176                ****************************************************************
177                **
178                ** Name:(S) HASH1   -  Indexed Jump Through A GOTO Table
179                ** Name:(S) HASH2   -  Indexed Jump Through A GOTO Table
180                **
181                ** Category:   EXCUTL
182                **
183                ** Purpose:
184                **      Jump into a table of GOTOs (or other 4-nibble beasts)
185                **      according to an index variable.
186                **
187                ** Entry:
188                **      A[A] = Hash byte (maximum 3FF).
189                **    HASH1: RSTK = Address of start of GOTO table.
190                **    HASH2: C[A] = Address of start of GOTO table.
191                **
192                ** Exit:
193                **      This routine exits by jumping to the A[A]'th entry
194                **      in the GOTO table.
195                **
196                ** Calls:      None.
197                **
198                ** Uses.......
199                **             A[X], C[A].
200                **
201                ** Stk lvls:   HASH1: 0.
202                **             HASH2: 1.
203                **
204                ** Detail:
205                **      Typical use:
206                **        GOSBVL =HASH1
207                **        GOTO   L0
208                **        GOTO   L1
209                **        GOTO   L2
210                **        GOTO   L3
211                **        GOTO   L4
212                **        GOTO   L5
213                **        GOTO   L6
214                **           .
215                **           .
216                **      The GOSBVL puts the address of the GOTO table on RSTK;
217                **      HASH1 peels it off.  Note that this GOSBVL is actually
218                **      acting like a GOTO; control will never return to the
219                **      code in the vicinity of the GOSBVL.
```

```
                ** 
                ** History:
                ** 
                **    Date     Programmer          Modification
                **  --------   ----------   ---------------------------------
                **             SA           Wrote
                **  10/17/83   NM           Attempted to document
                ** 
                ********************************************************************
                ********************************************************************
1B0A1 07        =HASH1  C=RSTK                POP TABLE BASE ADDRESS
1B0A3 A34       =HASH2  A=A+A  X              MULTIPLY INPUT BY 4
1B0A6 A34               A=A+A  X
1B0A9 C2                C=A+C  A              COMPUTE JUMP ADDRESS
1B0AB 06                RSTK=C                PUSH JUMP ADDRESS
1B0AD 01                RTN                   JUMP TO ROUTINE
                ********************************************************************
                ********************************************************************
                ** 
                ** Name:(S) STUFF   -  Fill Memory With Stuff Or 0's
                ** Name:(S) WIPOUT  -  Fill Memory With Stuff Or 0's
                ** 
                ** Category:   GENUTL
                ** 
                ** Purpose:
                **      Fill up memory will a pre-determined 16-nibble pattern
                **      (STUFF) or with zeroes (WIPOUT).
                ** 
                ** Entry:
                **      HEX mode.
                **      D1 @ start of area to be stuffed.
                **      C[A] = length of area to be stuffed (in nibs).
                **    STUFF: A[W] = pattern to be stuffed into memory.
                **      (WIPOUT presets A[W] to 0).
                ** 
                ** Exit:
                **      P=0.
                **      Carry clear.
                **      D1 pointing past last nibble stuffed.
                ** 
                ** Calls:      None.
                ** 
                ** Uses.......
                **              P,C,D1.  WIPOUT: A.
                ** 
                ** Stk lvls:   0
                ** 
                ** History:
                ** 
                **    Date     Programmer          Modification
                **  --------   ----------   ---------------------------------
                **             SA           Wrote
                **  10/17/83   NM           Attempted to document
                ** 
                ********************************************************************
```

```
275                ****************************************************************
276 1B0AF AF0  =WIPOUT A=0    W              CREATE ZERO REGISTER
277 1B0B2 CE   =STUFF  C=C-1  A              IS THERE ANYTHING TO WIPE OUT?
278 1B0B4 453          GOC    EXITWO         IF NOT, RETURN
279 1B0B7 80D0         P=C    0
280 1B0BB F6           CSR    A
281 1B0BD CE           C=C-1  A              ONLY PARTIAL WORD?
282 1B0BF 4D1          GOC    TAILWO         IF SO, GOTO TAILWO
283 1B0C2 1517  MOREWO DAT1=A W              WIPE OUT 16 NIBBLES
284 1B0C6 17F          D1=D1+ 16             MOVE TO NEXT WORD
285 1B0C9 A6E          C=C-1  B
286 1B0CC 55F          GONC   MOREWO         SPEED TRICK
287 1B0CF A2E          C=C-1  XS
288 1B0D2 5FE          GONC   MOREWO         SPEED TRICK
289 1B0D5 B36          C=C+1  X
290 1B0D8 CE           C=C-1  A              DONE WITH WHOLE WORDS?
291 1B0DA 57E          GONC   MOREWO         IF NOT, REPEAT MOREWO
292 1B0DD 1511  TAILWO DAT1=A WP             WIPE OUT PARTIAL WORD
293 1B0E1 137          CD1EX                 MOVE PAST PARTIAL WORD
294 1B0E4 809          C+P+1
295 1B0E7 137          CD1EX
296 1B0EA 20    EXITWO P=     0
297 1B0EC 03           RTNCC                 RETURN WITH CARRY CLEAR
298                ****************************************************************
299                ****************************************************************
300                **
301                ** Name:(S) MOVEDM  -  Blk Move To Higher Addr
302                ** Name:(S) MOVEDO  -  Blk Move To Higher Addr
303                ** Name:(S) MOVEDA  -  Blk Move To Higher Addr
304                ** Name:(S) MOVED1  -  Blk Move To Higher Addr
305                ** Name:(S) MOVED2  -  Blk Move To Higher Addr
306                ** Name:(S) MOVED3  -  Blk Move To Higher Addr
307                ** Name:(S) MOVEDD  -  Blk Move To Higher Addr
308                **
309                ** Category:   GENUTL
310                **
311                ** Purpose:
312                **      Block move of memory to higher address.
313                **
314                ** Entry:
315                **      MOVEDM: A[A] @ end of destination
316                **              B[A]   = block length
317                **              C[A] @ end of source
318                **
319                **      MOVEDO: D0 @ end of source
320                **              D1 @ end of destination
321                **              B[A]   = block length
322                **
323                **      MOVEDA: =AVMEME @ start of source
324                **              D1 @ end of destination
325                **              A[A] @ end of source
326                **
327                **      MOVED1: D0 @ pointer to start of source
328                **              D1 @ end of destination
329                **              A[A] @ end of source
```

```
330                **
331                **      MOVED2: D1 @ end of destination
332                **              A[A] @ end of source
333                **              C[A] @ start of source
334                **
335                **      MOVEDD: A[A] @ end of source
336                **              D1 @ end of destination
337                **              C[A] = block length
338                **
339                **      MOVED3: D0 @ end of source
340                **              D1 @ end of destination
341                **              C[A]   = block length
342                **
343                ** Exit:
344                **      P=0.
345                **      D0 @ start of source.
346                **      D1 @ start of destination.
347                **
348                ** Calls:      None.
349                **
350                ** Uses.......
351                **              A,C[A],D0,D1,P.
352                **
353                ** Stk lvls:   0
354                **
355                ** History:
356                **
357                **    Date      Programmer              Modification
358                **  --------    ----------   ----------------------------------
359                **              SA           Wrote
360                **  10/17/83    NM           Attempted to document
361                **
362                ********************************************************************
363                ********************************************************************
364 1B0EE 131   =MOVEDM D1=A
365 1B0F1 134         D0=C
366 1B0F4 D9    =MOVED0 C=B    A
367 1B0F6 6210        GOTO   MOVED3
368 1B0FA 1B00 =MOVEDA D0=(5) =AVMEME
          000
369 1B101 146   =MOVED1 C=DAT0 A
370 1B104 EE    =MOVED2 C=A-C  A
371 1B106 130   =MOVEDD D0=A
372 1B109 CE    =MOVED3 C=C-1  A              ANYTHING TO MOVE?
373 1B10B 4C4         GOC    EXITMD         IF NOT, RETURN
374 1B10E 80D0        P=C    0
375 1B112 F6          CSR    A
376 1B114 CE          C=C-1  A              ONLY PARTIAL WORD?
377 1B116 442         GOC    TAILMD         IF SO, GOTO TAILMD
378 1B119 18F   MOREMD D0=D0- 16             MOVE POINTERS TO NEXT WORD
379 1B11C 1CF         D1=D1- 16
380 1B11F 1527        A=DAT0 W               READ WORD
381 1B123 1517        DAT1=A W               WRITE WORD
382 1B127 A6E         C=C-1  B
383 1B12A 5EE         GONC   MOREMD          SPEED TRICK
```

```
 384 1B12D A2E          C=C-1  XS
 385 1B130 58E          GONC   MOREMD          SPEED TRICK
 386 1B133 B36          C=C+1  X
 387 1B136 CE           C=C-1  A               DONE WITH WHOLE WORDS?
 388 1B138 50E          GONC   MOREMD          IF NOT, REPEAT MOREMD
 389 1B13B D2    TAILMD C=0    A               MOVE POINTERS PAST PARTIAL WORD
 390 1B13D 809          C+P+1
 391 1B140 132          ADOEX
 392 1B143 EA           A=A-C  A
 393 1B145 130          DO=A
 394 1B148 133          AD1EX
 395 1B14B EA           A=A-C  A
 396 1B14D 131          D1=A
 397 1B150 1521         A=DATO WP              READ PARTIAL WORD
 398 1B154 1511         DAT1=A WP              WRITE PARTIAL WORD
 399 1B158 20    EXITMD P=     0
 400 1B15A 03           RTNCC                  RETURN WITH CARRY CLEAR
 401             ****************************************************************
 402             ****************************************************************
 403             **
 404             ** Name:(S) MOVEUM  -  Blk Move To Lower Addr
 405             ** Name:(S) MOVEU0  -  Blk Move To Lower Addr
 406             ** Name:(S) MOVEUA  -  Blk Move To Lower Addr
 407             ** Name:(S) MOVEU1  -  Blk Move To Lower Addr
 408             ** Name:(S) MOVEU2  -  Blk Move To Lower Addr
 409             ** Name:(S) MOVEU3  -  Blk Move To Lower Addr
 410             ** Name:(S) MOVEU4  -  Blk Move To Lower Addr
 411             **
 412             ** Category:   GENUTL
 413             **
 414             ** Purpose:
 415             **      Move a block of memory to a lower address.
 416             **
 417             ** Entry:
 418             **      MOVEUM: A[A] @ start of destination
 419             **              B[A]   = block length
 420             **              C[A] @ start of source
 421             **
 422             **      MOVEU0: D0 @ start of source
 423             **              D1 @ start of destination
 424             **              B[A]   = block length
 425             **
 426             **      MOVEUA: =AVMEMS @ end of source
 427             **              D1 @ start of destination
 428             **              A[A] @ start of source
 429             **
 430             **      MOVEU1: D0 @ pointer to end of source
 431             **              D1 @ start of destination
 432             **              A[A] @ start of source
 433             **
 434             **      MOVEU2: D1 @ start of destination
 435             **              A[A] @ start of source
 436             **              C[A] @ end of source
 437             **
 438             **      MOVEU3: D0 @ start of source
```

```
                **              D1 @ start of destination
                **              C[A]   = block length
                **
                **      MOVEU4: A[A] @ start of source
                **              D1 @ start of destination
                **              C[A] = block length
                **
                ** Exit:
                **      P=0.
                **      D0 @ end of source.
                **      D1 @ end of destination.
                **
                ** Calls:      None.
                **
                ** Uses.......
                **              A,C[A],D0,D1,P.
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------    ----------------------------------
                **              SA            Wrote
                **  10/17/83    NM            Attempted to document
                **
                ********************************************************************
                ********************************************************************
1B15C 131       =MOVEUM D1=A
1B15F 134               D0=C
1B162 D9        =MOVEU0 C=B     A
1B164 6210              GOTO    MOVEU3
1B168 1B00      =MOVEUA D0=(5)  =AVMEMS
      000
1B16F 146       =MOVEU1 C=DAT0  A
1B172 E2        =MOVEU2 C=C-A   A
1B174 130       =MOVEU4 D0=A
1B177 CE        =MOVEU3 C=C-1   A               ANYTHING TO MOVE?
1B179 494               GOC     EXITMU          IF NOT, RETURN
1B17C 80D0              P=C     0
1B180 F6                CSR     A
1B182 CE                C=C-1   A               ONLY PARTIAL WORD?
1B184 442               GOC     TAILMU          IF SO, GOTO TAILMU
1B187 1527  MOREMU      A=DAT0  W               READ WORD
1B18B 1517              DAT1=A  W               WRITE WORD
1B18F 16F               D0=D0+  16              MOVE POINTERS TO NEXT WORD
1B192 17F               D1=D1+  16
1B195 A6E               C=C-1   B
1B198 5EE               GONC    MOREMU          SPEED TRICK
1B19B A2E               C=C-1   XS
1B19E 58E               GONC    MOREMU          SPEED TRICK
1B1A1 B36               C=C+1   X
1B1A4 CE                C=C-1   A               DONE WITH WHOLE WORDS?
1B1A6 50E               GONC    MOREMU          IF NOT, REPEAT MOREMU
1B1A9 1521  TAILMU      A=DAT0  WP              READ PARTIAL WORD
```

```
493 1B1AD 1511          DAT1=A WP             WRITE PARTIAL WORD
494 1B1B1 136           CDOEX                 MOVE POINTERS PAST PARTIAL WORD
495 1B1B4 809           C+P+1
496 1B1B7 134           DO=C
497 1B1BA 137           CD1EX
498 1B1BD 809           C+P+1
499 1B1C0 135           D1=C
500 1B1C3 20    EXITMU  P=     0
501 1B1C5 03            RTNCC                 RETURN WITH CARRY CLEAR
502             ********************************************************************
503             ********************************************************************
504             **
505             ** Name:(S) STRTST  -  Test Strings For Equality
506             ** Name:(S) STREQL  -  Test Strings For Equality
507             **
508             ** Category:   GENUTL
509             **
510             ** Purpose:
511             **      Test two strings for equality.
512             **
513             ** Entry:
514             **   STRTST:
515             **      DO and D1 at high-memory end of the two strings to
516             **        be compared.
517             **      C[A] = block comparison length (in nibbles).
518             **   STREQL:
519             **      DO and D1 at high-memory end of the two strings to be
520             **        compared.
521             **      B[A] = (block comparison length - 1)/16.
522             **      P = (block comparison length - 1) mod 16.
523             **
524             ** Exit:
525             **      If comparison length = 0, carry clear and XM=1.
526             **      If strings equal, carry clear and XM=0.
527             **      If strings not equal, carry set and XM=0.
528             **      P can be anything.
529             **      B[A] contains remnant of length/16.
530             **      A, C contains first words not equal.
531             **      DO and D1 point at first words not equal.
532             **
533             ** Calls:      None.
534             **
535             ** Uses.......
536             **              A,B[A],C,P,DO,D1.
537             **
538             ** Stk lvls:   0
539             **
540             ** History:
541             **
542             **    Date      Programmer              Modification
543             **  --------    ----------   ------------------------------------
544             **              SA           Wrote
545             **  10/18/83    NM           Attempted to document
546             **
547             ********************************************************************
```

```
548                *****************************************************************
549 1B1C7 821      =STRTST XM=0
550 1B1CA CE               C=C-1  A             ANYTHING TO TEST?
551 1B1CC 415              GOC    EXITST        IF NOT, GOTO EXITST
552 1B1CF 80D0             P=C    0
553 1B1D3 F6               CSR    A
554 1B1D5 D5               B=C    A
555 1B1D7 CD               B=B-1  A             ONLY PARTIAL WORD?
556 1B1D9 4A1              GOC    TAILST        IF SO, GOTO TAILST
557 1B1DC 18F     MOREST   D0=D0- 16            MOVE POINTERS TO NEXT WORDS
558 1B1DF 1CF              D1=D1- 16
559 1B1E2 1527             A=DAT0 W             READ WORDS
560 1B1E6 1577             C=DAT1 W
561 1B1EA 976              ?A#C   W             EQUAL?
562 1B1ED 00               RTNYES               IF NOT, RETURN CARRY SET, XM=0
563 1B1EF CD      =STREQL  B=B-1  A             DONE WITH WHOLE WORDS?
564 1B1F1 5AE              GONC   MOREST        IF NOT, REPEAT MOREST
565 1B1F4 D2      TAILST   C=0    A             MOVE POINTERS TO PARTIAL WORD
566 1B1F6 809              C+P+1
567 1B1F9 132              ADOEX
568 1B1FC EA               A=A-C  A
569 1B1FE 132              ADOEX
570 1B201 133              AD1EX
571 1B204 EA               A=A-C  A
572 1B206 133              AD1EX
573 1B209 AF0              A=0    W             READ PARTIAL WORDS
574 1B20C 1521             A=DAT0 WP
575 1B210 AF2              C=0    W
576 1B213 1571             C=DAT1 WP
577 1B217 916              ?A#C   WP            EQUAL?
578 1B21A 00               RTNYES               IF NOT, RETURN CARRY SET, XM=0
579 1B21C 03               RTNCC                RETURN CARRY CLEAR, XM=0
580 1B21E A6E     EXITST   C=C-1  B             CLEAR CARRY
581 1B221 00               RTNSXM               RETURN CARRY CLEAR, XM=1
582                *****************************************************************
583                *****************************************************************
584                **
585                ** Name:(S) FLTDH   -  Convert 12-digit Flt To Hex Integer
586                ** Name:(S) DCHXF   -  Convert 12-digit Flt To Hex Integer
587                **
588                ** Category:   CONVRT
589                **
590                ** Purpose:
591                **      Convert a 12-digit floating-point number to a 5-digit
592                **      hex integer.
593                **
594                ** Entry:
595                **      A=12-digit floating-point number.
596                **      (FLTDH and DCHXF are two names for same entry point.)
597                **
598                ** Exit:
599                **      P=0.
600                **      A[A] = hex integer.
601                **      Carry set if number is positive and in range.
602                **      Carry clear ->
```

```
603             **        If XM=1, number is out of range (returns FFFFF).
604             **          (NaN is considered out-of-range.)
605             **        If XM=0, number is negative (returns result in 2's
606             **          complement).
607             **      Also B[S]#0 iff number is negative.
608             **      HEX mode.
609             **
610             ** Calls:      OVFLOW.
611             **
612             ** Uses.......
613             **             A,B,C,P,XM.
614             **
615             ** Stk lvls:   1
616             **
617             ** History:
618             **
619             **    Date      Programmer              Modification
620             **  --------    ----------   ---------------------------------
621             **              SA           Wrote
622             **  12/20/82    SW           Added info about B[S] to doc hdr
623             **  10/18/83    NM           Attempted to document
624             **
625             ****************************************************************
626             ****************************************************************
627 1B223      =DCHXF
628 1B223 04   =FLTDH  SETHEX                INITIALIZE
629 1B225 821          XM=0
630 1B228 20           P=      0
631 1B22A AC8          B=A     S              SAVE SIGN
632 1B22D D1           B=0     A
633 1B22F B05          B=B+1   P
634 1B232 3260         LCHEX   006
        0
635 1B237 9B6          ?A>C    X              EXPONENT TOO LARGE?
636 1B23A 76           GOYES   FLTDH2         IF SO, GOTO FLTDH2
637 1B23C B8C          A=-A-1  P              LOCATE FIRST FRACTIONAL DIGIT
638 1B23F A86          C=A     P
639 1B242 80D0         P=C     0
640 1B246 0D           P=P-1
641 1B248 BF0          ASL     W              ALIGN MANTISSA
642 1B24B 05           SETDEC
643 1B24D A04          A=A+A   P              ROUND DOWN?
644 1B250 A90          A=0     WP
645 1B253 04           SETHEX
646 1B255 563          GONC    DHBGN          IF SO, GOTO DHBGN
647 1B258 05           SETDEC
648 1B25A A1C          A=A-1   WP             ROUND UP
649 1B25D B74          A=A+1   W              DECADE ROLLOVER?
650 1B260 04           SETHEX
651 1B262 592          GONC    DHBGN          IF NOT, GOTO DHBGN
652 1B265 B44          A=A+1   S              CREATE UNIT IN NEXT DECADE
653 1B268 0D           P=P-1
654 1B26A 887          ?P#     7              OVERFLOW?
655 1B26D F1           GOYES   DHBGN          IF NOT, GOTO DHBGN
656 1B26F 7511 FLTDH1  GOSUB   OVFLOW         CREATE FFFFF, CLEAR CARRY
```

```
1B273 20          P=     0
1B275 00          RTNSXM                 RETURN EXTERNAL MODULE MISSING
1B277 C0    DHADD ADD    A,B  A
1B279 45F         GOC    FLTDH1
1B27C A0C   DHDCR A=A-1  P
1B27F 57F         GONC   DHADD
1B282 C5          B=B+B  A               NEXT DECADE
1B284 D9          C=B    A
1B286 C5          B=B+B  A
1B288 C5          B=B+B  A
1B28A C1          B=B+C  A
1B28C 0C    DHBGN P=P+1                  CONVERSION COMPLETE?
1B28E 5DE         GONC   DHDCR           IF NOT, REPEAT DHDCR
1B291 04    DHSGN SETHEX
1B293 949         ?B=0   S               POSITIVE NUMBER?
1B296 00          RTNYES                 IF SO, RETURN CARRY SET
1B298 8A8         ?A=0   A               ZERO?
1B29B 00          RTNYES                 IF SO, RETURN CARRY SET
1B29D F8          A=-A   A               NEGATE
1B29F 03          RTNCC                  RETURN CARRY CLEAR
1B2A1 3200  FLTDH2 LCHEX  F00
          F
1B2A6 922         ?A=C   XS              IEEE CRITTER?
1B2A9 6C          GOYES  FLTDH1          IF SO, GOTO FLTDH1
1B2AB 05          SETDEC
1B2AD AA6         C=A    XS
1B2B0 B34         A=A+1  X               EXPONENT = -1?
1B2B3 D0          A=0    A
1B2B5 5E0         GONC   FLTDH3          IF NOT, GOTO FLTDH3
1B2B8 A54         A=A+A  M
1B2BB 55D         GONC   DHSGN
1B2BE E4          A=A+1  A
1B2C0 60DF        GOTO   DHSGN           GOTO DHSGN
1B2C4 A26   FLTDH3 C=C+C  XS             NEGATIVE EXPONENT?
1B2C7 49C         GOC    DHSGN           IF SO, GOTO DHSGN
1B2CA 04          SETHEX
1B2CC 62AF        GOTO   FLTDH1          GOTO FLTDH1
              ****************************************************************
              ****************************************************************
              **
              ** Name:(S) DECHEX  -  Convert DEC Integer To HEX Integer
              ** Name:(S) DCHX=C  -  Convert DEC Integer To HEX Integer
              **
              ** Category:   CONVRT
              **
              ** Purpose:
              **      Convert decimal integer to hex integer.
              **
              ** Entry:
              **   DECHEX: A[W] = decimal integer.
              **   DCHX=C: C[W] = decimal integer.
              **
              ** Exit:
              **      P=0.
              **      A[A] = hex integer.
```

```
711             **      HEX mode.
712             **      Carry set -> result is good.
713             **      Carry clear -> overflow.
714             **      XM = not carry.
715             **
716             ** Calls:      None.
717             **
718             ** Uses.......
719             **              A,B,C,P,XM.
720             **
721             ** Stk lvls:    1.
722             **
723             ** History:
724             **
725             **    Date      Programmer              Modification
726             **  --------    ----------    ----------------------------------
727             **              SA            Wrote
728             **  10/18/83    NM            Attempted to document
729             **
730             ********************************************************************
731             ********************************************************************
732 1B2D0 DA   =DCHX=C A=C     A
733 1B2D2 04   =DECHEX SETHEX                 INITIALIZE
734 1B2D4 821          XM=0
735 1B2D7 26           P=      6
736 1B2D9 A91          B=0     WP
737 1B2DC 20           P=      0
738 1B2DE B05          B=B+1   P
739 1B2E1 AF2          C=0     W
740 1B2E4 A86          C=A     P
741 1B2E7 A80   TEST   A=0     P
742 1B2EA 978          ?A=0    W               FINISHED?
743 1B2ED 32           GOYES   DONE            IF SO, GOTO DONE
744 1B2EF 06           RSTK=C
745 1B2F1 A15          B=B+B   WP              NEXT DECADE
746 1B2F4 A99          C=B     WP
747 1B2F7 A15          B=B+B   WP
748 1B2FA A15          B=B+B   WP
749 1B2FD A11          B=B+C   WP
750 1B300 07           C=RSTK
751 1B302 0C           P=P+1
752 1B304 A0C   RPTDH  A=A-1   P               DONE WITH DECADE?
753 1B307 4FD          GOC     TEST            IF SO, GOTO TEST
754 1B30A A19          C=B+C   WP              ADD ONE COUNT IN DECADE
755 1B30D 56F          GONC    RPTDH           UNCONDITIONAL REPEAT RPTDH
756 1B310 DE    DONE   ACEX    A               LEAVE RESULT IN A
757 1B312 20           P=      0
758 1B314 97A          ?C=0    W               OVERFLOW?
759 1B317 00           RTNYES                  IF NOT, RETURN CARRY SET
760 1B319 00           RTNSXM                  RETURN EXTERNAL MODULE MISSING
761             ********************************************************************
762             ********************************************************************
763             **
764             ** Name:(S) HDFLT   -  Convert HEX Integer To DEC Flt-pt
765             **
```

```
                ** Category:   CONVRT
                **
                ** Purpose:
                **      Convert hex integer to 12-digit decimal floating-point
                **      number.
                **
                ** Entry:
                **      A[A] = hex integer.
                **
                ** Exit:
                **      P=0.
                **      A=12-digit floating-point number.
                **      Carry set.
                **      DEC mode.
                **
                ** Calls:      HEXDEC.
                **
                ** Uses.......
                **              A,B,C,P.
                **
                ** Stk lvls:   1
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **   --------   ----------   ----------------------------------
                **              SA           Wrote
                **   10/15/82   SA           Changed to NM's conversion
                **   10/18/83   NM           Attempted to document
                **
                ********************************************************************
                ********************************************************************
1B31B 8F00 =HDFLT  GOSBVL =HEXDEC        CONVERT HEX INTEGER TO DECIMAL
      000
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) FLOAT   -  Convert Dec Integer Into 12-Dig Float
                **
                ** Category:   CONVRT
                **
                ** Purpose:
                **      Convert right-justified decimal integer into floating
                **      point number.
                **
                ** Entry:
                **      Argument in A[W] (unsigned).
                **      Maximum 999999999999 (1e12-1).
                **
                ** Exit:
                **      Floating-point number in A[W].
                **      DEC mode.
                **      Carry set.
                **
                ** Calls:      None.
```

```
                **
                ** Uses.......
                **            A[W], P.
                **
                ** Stk lvls:   0
                **
                ** Algorithm:
                **      Return if A=0.
                **      ASL 3 times, A[X]=011.
                **      While A[14]=0 do
                **        begin
                **          ASL M    {loop to align mantissa}
                **          A=A-1 X
                **        end.
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ----------------------------------
                **              SR           Wrote
                **  06/11/82    NM           Attempted to document
                **
                ********************************************************************
                ********************************************************************
                *
                * Note: Above code falls into this
                *
1B322 05        =FLOAT  SETDEC
1B324 978               ?A=0    W               ANYTHING TO MOVE?
1B327 00                RTNYES                  IF NOT, RETURN CARRY SET
1B329 BF0               ASL     W               MOVE MANTISSA TO M-FIELD
1B32C BF0               ASL     W
1B32F E4                A=A+1   A               CREATE EXPONENT OF 11
1B331 BF0               ASL     W
1B334 E4                A=A+1   A
1B336 2E                P=      14
1B338 570               GONC    FLTD20          UNCONDITIONAL GOTO FLTD20
1B33B BD0    FLTD10     ASL     M               SHIFT MANTISSA LEFT
1B33E CC                A=A-1   A               DECREMENT EXPONENT
1B340 908    FLTD20     ?A=0    P               SHIFT COMPLETE?
1B343 8F                GOYES   FLTD10          IF NOT, REPEAT FLTD10
1B345 20                P=      0
1B347 02                RTNSC                   RETURN CARRY SET
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) A-MULT  -  Multiply Two 20-bit Hex Integers
                **
                ** Category:   MTHUTL
                **
                ** Purpose:
                **      Multiply two 20-bit hex integers.
                **
                ** Entry:
                **      A[A], C[A] are operands.
```

```
875                **
876                ** Exit:
877                **      P preserved.
878                **      A[A]=product.
879                **      Carry set if no problem.
880                **      Carry clear -> overflow.  Returns FFFFF.
881                **
882                ** Calls:      None.
883                **
884                ** Uses.......
885                **             A[A],B[A],C[A],C[14].
886                **
887                ** Stk lvls:   0
888                **
889                **    Date     Programmer          Modification
890                **  --------   ----------  ----------------------------------
891                **             SA          Created
892                **  10/18/83   NM          Attempted to document
893                **
894                ******************************************************************
895                ******************************************************************
896 1B349 D8   =A-MULT B=A    A
897 1B34B D0           A=0    A
898 1B34D 8A9          ?B=0   A              ZERO MULTIPLIER?
899 1B350 00           RTNYES                IF SO, RETURN CARRY SET
900 1B352 80FE         CPEX   14
901 1B356 20           P=     0
902 1B358 5F0          GONC   AMULT2         GOTO AMULT2
903 1B35B F5    AMULT1 BSR    A              NEXT DIGIT
904 1B35D 80D4         P=C    4
905 1B361 880          ?P#    0              OVERFLOW?
906 1B364 02           GOYES  AMULT5         IF SO, GOTO AMULT5
907 1B366 F2           CSL    A              SHIFT MULTIPLICAND LEFT
908 1B368 B89   AMULT2 B=-B   P              ZERO DIGIT?
909 1B36B 5FE          GONC   AMULT1         IF SO, REPEAT AMULT1
910 1B36E CA    AMULT3 A=A+C  A              ADD MULTIPLICAND TO RESULT
911 1B370 431          GOC    AMULT5         ON OVERFLOW, GOTO AMULT5
912 1B373 B05          B=B+1  P              DONE WITH DIGIT?
913 1B376 57F          GONC   AMULT3         IF NOT, REPEAT AMULT3
914 1B379 8AD   AMULT4 ?B#0   A              MULTIPLICATION COMPLETE?
915 1B37C FD           GOYES  AMULT1         IF NOT, REPEAT AMULT1
916 1B37E 80DE         P=C    14             RESTORE REGISTER POINTER
917 1B382 02           RTNSC                 RETURN CARRY SET
918 1B384 80DE  AMULT5 P=C    14
919 1B388 D0    OVFLOW A=0    A
920 1B38A CC           A=A-1  A
921 1B38C 03           RTNCC
922                ******************************************************************
923                ******************************************************************
924                **
925                ** Name:(S) REV$    -  Reverse Characters In A String On Stack
926                **
927                ** Category:   MTHSTK
928                **
929                ** Purpose:
```

```
                **      Reverse a string on the mathstack.
                **
                ** Entry:
                **      HEX mode.
                **      D1 pointing at string header.
                **
                ** Exit:
                **      D1 pointing at string header.
                **      String has been reversed.
                **      C[A]=D[A]=copy of D0.
                **      Error exit (eDATTY) if not pointing at string.
                **
                ** Calls:      POP1S.
                **
                ** Uses.......
                **              A,B,C,D,P.
                **
                ** Stk lvls:   1
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ----------------------------------
                **              SA           Wrote
                **  10/18/83    NM           Attempted to document
                **
                ********************************************************************
                ********************************************************************
1B38E 8F00 =REV$    GOSBVL =POP1S          READ HEADER
           000
1B395 AF1            B=0    W
1B398 D8             B=A    A
1B39A 822            SB=0
1B39D 81D            BSRB
1B3A0 81D            BSRB
1B3A3 137            CD1EX
1B3A6 C9             C=B+C  A               COMPUTE ADDR OF MIDDLE OF STRING
1B3A8 C9             C=B+C  A
1B3AA 135            D1=C
1B3AD 136            CD0EX
1B3B0 D7             D=C    A
1B3B2 832            ?SB=0                  EVEN NUMBER OF CHARS IN STRING?
1B3B5 71             GOYES  EVEN            IF SO, GOTO EVEN
1B3B7 511            GONC   ODD             UNCONDITIONAL GOTO ODD
1B3BA 1C1   RPTREV   D1=D1- 2
1B3BD 14A            A=DAT0 B               READ SYMMETRIC CHARACTERS
1B3C0 14F            C=DAT1 B
1B3C3 149            DAT1=A B               WRITE REVERSED
1B3C6 14C            DAT0=C B
1B3C9 161   ODD      D0=D0+ 2
1B3CC CD    EVEN     B=B-1  A               DONE?
1B3CE 5BE            GONC   RPTREV          IF NOT, REPEAT RPTREV
1B3D1 1CF            D1=D1- 16              RESTORE POINTERS
1B3D4 DB             C=D    A
1B3D6 134            D0=C
```

```
 984 1B3D9 01          RTN                     RETURN
 985            ****************************************************************
 986            ****************************************************************
 987            **
 988            ** Name:(S) POPMTH  -  Skip Past An Item On Mthstk
 989            ** Name:(S) POPSTR  -  Skip Past An Item On Mthstk
 990            **
 991            ** Category:   MTHSTK
 992            **
 993            ** Purpose:
 994            **      Skip past current item on the mathstack.  Useful for
 995            **      finding a particular item or for counting items.
 996            **
 997            ** Entry:
 998            **      P=0.
 999            **   POPMTH: D1 at top of mathstack.
1000            **   POPSTR: D1 pointing past first 2 nibbles of string header
1001            **           at top of mathstack.
1002            **
1003            ** Exit:
1004            **      P=0.
1005            **      D1 at new top of mathstack.
1006            **      Carry clear.
1007            **
1008            ** Calls:      None.
1009            **
1010            ** Uses.......
1011            **             A,C,D1.
1012            **
1013            ** Stk lvls:   0
1014            **
1015            ** Detail:
1016            **      Correctly skips past complex numbers and string items.
1017            **
1018            ** History:
1019            **
1020            **    Date      Programmer              Modification
1021            **  --------    ----------   ----------------------------------
1022            **              SC           Wrote
1023            **  10/18/83    NM           Attempted to document
1024            **
1025            ****************************************************************
1026            ****************************************************************
1027 1B3DB 30A  =POPMTH LCHEX  A
1028 1B3DE 14B          A=DAT1 B
1029 1B3E1 982          ?A<C   P               IS IT A REAL NUMBER ?
1030 1B3E4 11           GOYES  POPS10          IF SO, GOTO POPS10
1031 1B3E6 171          D1=D1+ 2
1032 1B3E9 31E0         LCHEX  0E
1033 1B3ED 966          ?A#C   B               COMPLEX NUMBER?
1034 1B3F0 A0           GOYES  POPS20          IF NOT, GOTO POPS20
1035 1B3F2 17F          D1=D1+ 16
1036 1B3F5 17F   POPS10 D1=D1+ 16
1037 1B3F8 03           RTNCC                  RETURN CARRY CLEAR
1038 1B3FA B04   POPS20 A=A+1  P
```

```
1039 1B3FD A64          A=A+A  B
1040 1B400 96C          ?A#0   B              IS IT A STRING ?
1041 1B403 D0           GOYES  POPS30         IF NOT, GOTO POPS30
1042 1B405 143  =POPSTR A=DAT1 A              READ THE STRING LENGTH
1043 1B408 137          CD1EX
1044 1B40B C2           C=A+C  A              COMPUTE ADDRESS OF NEXT ELEMENT
1045 1B40D 137          CD1EX
1046 1B410 17D   POPS30 D1=D1+ 14
1047 1B413 03           RTNCC                 RETURN CARRY CLEAR
1048            ********************************************************************
1049            ********************************************************************
1050            **
1051            ** Name:(S) CSRC1   -  Perform 1 CSRC
1052            ** Name:(S) CSRC2   -  Perform 2 CSRCs
1053            ** Name:(S) CSRC3   -  Perform 3 CSRCs
1054            ** Name:(S) CSRC4   -  Perform 4 CSRCs
1055            ** Name:(S) CSRC5   -  Perform 5 CSRCs
1056            ** Name:(S) CSRC6   -  Perform 6 CSRCs
1057            ** Name:(S) CSRC7   -  Perform 7 CSRCs
1058            ** Name:(S) CSRC8   -  Perform 8 CSRCs
1059            ** Name:(S) CSRC9   -  Perform 9 CSRCs
1060            ** Name:(S) CSRC10  -  Perform 10 CSRCs
1061            ** Name:(S) CSRC11  -  Perform 11 CSRCs
1062            ** Name:(S) CSRC12  -  Perform 12 CSRCs
1063            ** Name:(S) CSRC13  -  Perform 13 CSRCs
1064            ** Name:(S) CSRC14  -  Perform 14 CSRCs
1065            ** Name:(S) CSRC15  -  Perform 15 CSRCs
1066            ** Name:(S) CSLC1   -  Perform 1 CSLC
1067            ** Name:(S) CSLC2   -  Perform 2 CSLCs
1068            ** Name:(S) CSLC3   -  Perform 3 CSLCs
1069            ** Name:(S) CSLC4   -  Perform 4 CSLCs
1070            ** Name:(S) CSLC5   -  Perform 5 CSLCs
1071            ** Name:(S) CSLC6   -  Perform 6 CSLCs
1072            ** Name:(S) CSLC7   -  Perform 7 CSLCs
1073            ** Name:(S) CSLC8   -  Perform 8 CSLCs
1074            ** Name:(S) CSLC9   -  Perform 9 CSLCs
1075            ** Name:(S) CSLC10  -  Perform 10 CSLCs
1076            ** Name:(S) CSLC11  -  Perform 11 CSLCs
1077            ** Name:(S) CSLC12  -  Perform 12 CSLCs
1078            ** Name:(S) CSLC13  -  Perform 13 CSLCs
1079            ** Name:(S) CSLC14  -  Perform 14 CSLCs
1080            ** Name:(S) CSLC15  -  Perform 15 CSLCs
1081            **
1082            ** Category:   GENUTL
1083            **
1084            ** Purpose:
1085            **      Perform 1 to 15 circular left or right shifts to C.
1086            **
1087            ** Entry:
1088            **      None.
1089            **
1090            ** Exit:
1091            **      C-register shifted.
1092            **
1093            ** Calls:      None.
```

```
1094             **
1095             ** Uses.......
1096             **              C.
1097             **
1098             ** Stk lvls:    0
1099             **
1100             ** History:
1101             **
1102             **     Date      Programmer              Modification
1103             **   --------    ----------   ------------------------------------
1104             **                 SA          Wrote
1105             **   10/18/83      NM          Attempted to document
1106             **
1107             ********************************************************************
1108             ********************************************************************
1109 1B415       =CSLC9
1110 1B415 816   =CSRC7  CSRC
1111 1B418       =CSLC10
1112 1B418 816   =CSRC6  CSRC
1113 1B41B       =CSLC11
1114 1B41B 816   =CSRC5  CSRC
1115 1B41E       =CSLC12
1116 1B41E 816   =CSRC4  CSRC
1117 1B421       =CSLC13
1118 1B421 816   =CSRC3  CSRC
1119 1B424       =CSLC14
1120 1B424 816   =CSRC2  CSRC
1121 1B427       =CSLC15
1122 1B427 816   =CSRC1  CSRC
1123 1B42A 01            RTN
1124             ********************************************************************
1125 1B42C       =CSRC8
1126 1B42C 812   =CSLC8  CSLC
1127 1B42F       =CSRC9
1128 1B42F 812   =CSLC7  CSLC
1129 1B432       =CSRC10
1130 1B432 812   =CSLC6  CSLC
1131 1B435       =CSRC11
1132 1B435 812   =CSLC5  CSLC
1133 1B438       =CSRC12
1134 1B438 812   =CSLC4  CSLC
1135 1B43B       =CSRC13
1136 1B43B 812   =CSLC3  CSLC
1137 1B43E       =CSRC14
1138 1B43E 812   =CSLC2  CSLC
1139 1B441       =CSRC15
1140 1B441 812   =CSLC1  CSLC
1141 1B444 01            RTN
1142 1B446               END
```

```
=A-MULT  Abs  111433 #1B349 -   896
 AMULT1  Abs  111451 #1B35B -   903   909   915
 AMULT2  Abs  111464 #1B368 -   908   902
 AMULT3  Abs  111470 #1B36E -   910   913
 AMULT4  Abs  111481 #1B379 -   914
 AMULT5  Abs  111492 #1B384 -   918   906   911
 AR00    Abs  110638 #1B02E -    58    55
 AR05    Abs  110640 #1B030 -    59    57
 AR10    Abs  110646 #1B036 -    61
 AR17    Abs  110683 #1B05B -    74    71
 AR20    Abs  110686 #1B05E -    75    63    67
 AR25    Abs  110696 #1B068 -    79    53    77
 AR30    Abs  110700 #1B06C -    81    73
 AVMEME  Ext              -   368
 AVMEMS  Ext              -   471
=CSLC1   Abs  111681 #1B441 -  1140
=CSLC10  Abs  111640 #1B418 -  1111
=CSLC11  Abs  111643 #1B41B -  1113
=CSLC12  Abs  111646 #1B41E -  1115
=CSLC13  Abs  111649 #1B421 -  1117
=CSLC14  Abs  111652 #1B424 -  1119
=CSLC15  Abs  111655 #1B427 -  1121
=CSLC2   Abs  111678 #1B43E -  1138
=CSLC3   Abs  111675 #1B43B -  1136
=CSLC4   Abs  111672 #1B438 -  1134
=CSLC5   Abs  111669 #1B435 -  1132
=CSLC6   Abs  111666 #1B432 -  1130
=CSLC7   Abs  111663 #1B42F -  1128
=CSLC8   Abs  111660 #1B42C -  1126
=CSLC9   Abs  111637 #1B415 -  1109
=CSRC1   Abs  111655 #1B427 -  1122
=CSRC10  Abs  111666 #1B432 -  1129
=CSRC11  Abs  111669 #1B435 -  1131
=CSRC12  Abs  111672 #1B438 -  1133
=CSRC13  Abs  111675 #1B43B -  1135
=CSRC14  Abs  111678 #1B43E -  1137
=CSRC15  Abs  111681 #1B441 -  1139
=CSRC2   Abs  111652 #1B424 -  1120
=CSRC3   Abs  111649 #1B421 -  1118
=CSRC4   Abs  111646 #1B41E -  1116
=CSRC5   Abs  111643 #1B41B -  1114
=CSRC6   Abs  111640 #1B418 -  1112
=CSRC7   Abs  111637 #1B415 -  1110
=CSRC8   Abs  111660 #1B42C -  1125
=CSRC9   Abs  111663 #1B42F -  1127
=DCHX=C  Abs  111312 #1B2D0 -   732
=DCHXF   Abs  111139 #1B223 -   627
=DECHEX  Abs  111314 #1B2D2 -   733
 DHADD   Abs  111223 #1B277 -   659   662
 DHBGN   Abs  111244 #1B28C -   668   646   651   655
 DHDCR   Abs  111228 #1B27C -   661   669
 DHSGN   Abs  111249 #1B291 -   670   686   688   690
 DONE    Abs  111376 #1B310 -   756   743
=DRANGE  Abs  110710 #1B076 -   124
 EVEN    Abs  111564 #1B3CC -   979   971
```

```
 EXITMD  Abs  110936 #1B158 -   399   373
 EXITMU  Abs  111043 #1B1C3 -   500   476
 EXITST  Abs  111134 #1B21E -   580   551
 EXITWO  Abs  110826 #1B0EA -   296   278
=FLOAT   Abs  111394 #1B322 -   847
 FLTD10  Abs  111419 #1B33B -   857   860
 FLTD20  Abs  111424 #1B340 -   859   856
=FLTDH   Abs  111139 #1B223 -   628
 FLTDH1  Abs  111215 #1B26F -   656   660   679   692
 FLTDH2  Abs  111265 #1B2A1 -   677   636
 FLTDH3  Abs  111300 #1B2C4 -   689   684
=HASH1   Abs  110753 #1B0A1 -   230
=HASH2   Abs  110755 #1B0A3 -   231
=HDFLT   Abs  111387 #1B31B -   798
 HEXDEC  Ext                -   798
 MBR10   Abs  110731 #1B08B -   166   172
 MBR20   Abs  110749 #1B09D -   173   170
=MEMBER  Abs  110744 #1B098 -   171
 MOREMD  Abs  110873 #1B119 -   378   383   385   388
 MOREMU  Abs  110983 #1B187 -   481   486   488   491
 MOREST  Abs  111068 #1B1DC -   557   564
 MOREWO  Abs  110786 #1B0C2 -   283   286   288   291
=MOVED0  Abs  110836 #1B0F4 -   366
=MOVED1  Abs  110849 #1B101 -   369
=MOVED2  Abs  110852 #1B104 -   370
=MOVED3  Abs  110857 #1B109 -   372   367
=MOVEDA  Abs  110842 #1B0FA -   368
=MOVEDD  Abs  110854 #1B106 -   371
=MOVEDM  Abs  110830 #1B0EE -   364
=MOVEU0  Abs  110946 #1B162 -   469
=MOVEU1  Abs  110959 #1B16F -   472
=MOVEU2  Abs  110962 #1B172 -   473
=MOVEU3  Abs  110967 #1B177 -   475   470
=MOVEU4  Abs  110964 #1B174 -   474
=MOVEUA  Abs  110952 #1B168 -   471
=MOVEUM  Abs  110940 #1B15C -   467
 ODD     Abs  111561 #1B3C9 -   978   972
 OVFLOW  Abs  111496 #1B388 -   919   656
 POP1S   Ext                -   958
=POPMTH  Abs  111579 #1B3DB -  1027
 POPS10  Abs  111605 #1B3F5 -  1036  1030
 POPS20  Abs  111610 #1B3FA -  1038  1034
 POPS30  Abs  111632 #1B410 -  1046  1041
=POPSTR  Abs  111621 #1B405 -  1042
=RANGE   Abs  110716 #1B07C -   125
=REV$    Abs  111502 #1B38E -   958
=RND-12  Abs  110623 #1B01F -    52
 RPTDH   Abs  111364 #1B304 -   752   755
 RPTREV  Abs  111546 #1B3BA -   973   980
=STREQL  Abs  111087 #1B1EF -   563
=STRTST  Abs  111047 #1B1C7 -   549
=STUFF   Abs  110770 #1B0B2 -   277
 TAILMD  Abs  110907 #1B13B -   389   377
 TAILMU  Abs  111017 #1B1A9 -   492   480
 TAILST  Abs  111092 #1B1F4 -   565   556
```

```
 TAILWO  Abs  110813 #1B0DD -   292   282
 TEST    Abs  111335 #1B2E7 -   741   753
=WIPOUT  Abs  110767 #1B0AF -   276
```

Input Parameters

  Source file name is AB&UTL::MS

  Listing file name is AB/UTL:TI:ML::-1

  Object file name is AB%UTL:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1              *
 2              *     M   M  BBBB    &     III  M   M   GGG
 3              *     MM MM  B   B  & &     I   MM MM  G   G
 4              *     M M M  B   B  & &     I   M M M  G
 5              *     M M M  BBBB    &      I   M M M  G GGG
 6              *     M   M  B   B  & & &   I   M   M  G   G
 7              *     M   M  B   B  &  &    I   M   M  G   G
 8              *     M   M  BBBB    && &  III  M   M   GGG
 9
10              *
11                       TITLE  IMAGE Parse Routines <831212.1518>
12 1B446                 ABS    #1B446
13              *
14              ************************************************************
15              *   FIXED ENTRY POINTS (labels which are not used in
16              *      the HP-71 mainframe, but which should be kept
17              *      as external entry points for applications):
18              *         CkLoop
19              *         CkLpNC
20              *         BOPNM-
21              *         BOPW05
22              *         IMinit
23              *         SetAVM
24              *         CSL9R0
25              *         IMD0-2
26              *         IMD0+2
27              *         D12R0A
28              *         IMoffs
29              *         BldIMA
30              *         BldIMG
31              *         PRSsc+
32              *         PRSscn
33              *
```

```
                EJECT
*********************************************************
*  These following symbol values and relationships      *
*  are used in the IMAGE routines:                      *
*       tLBLRF = (tLINE#)-1                             *
*       eSTMNF = (eLN#NF)+1                             *
*       tIMAGE = FF                                     *
*********************************************************
*--- Image tokens for building expanded IMAGE.
** 1) Tokens not identifying the end of a numeric field.
**   1a) Tokens not used in backwards search.
=uSTRPT EQU    #D0      String pointer
=uMULT  EQU  (uSTRPT)+1 |D1| Multiplier
=uLOOPB EQU    #D2      Loop on byte
=uLOOPS EQU    #D3      Loop on string (12 nibs)
=uIMXCH EQU    #D4      Strange execution character.
*
**   1b) Tokens used in backwards search.
=uOPNNM EQU    #D8      Open loop without multiplier
=uJMP{} EQU    #D9      Jump over paren loop ptr (9 nibs)
=uJMPst EQU    #DA      Jump over string pointer (14 nibs)
=uJMPdl EQU    #DB      Jump over unfilled delimiter (8nibs)
=uIMbck EQU    #DC      Poll for backward search handler
=uIMsta EQU    #DE      IMAGE string start (|Dx| - see IMentr)
=uOPNM- EQU    #DF      Open loop with mult, decremented
=uOPNWM EQU  (uOPNM-)+1 |E0| Open loop with mult  (ends in 0!)
*
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
=EndNum EQU    #E6      Any value >= this identifies the    +
*+                      end of a numeric field (used        +
*+                      in execution).                      +
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*
** 2) Tokens identifying the end of a numeric field.
**   2a) Tokens not used in backwards search.
=uCPLXC EQU    #EE      Complex field closed
=uLOOPP EQU    #EF      Loop on parentheses (variable #bytes)
=uIMend EQU    #F0      |F0| IMAGE string end
*
**   2b) Tokens used in backwards search.
=uRESTP EQU    #F1      Restart parse
=uDELIM EQU    #F4      Delimiter
**       Tokens delimiting an output/input field.
=uHKB^  EQU    #F6      H,K,B or ^ field
=uALit  EQU    #F7      "A" literal field
=uNUMNn EQU    #F8      |F8| Numeric, no float chars, no sign*
=uNUMNs EQU  (uNUMNn)+1 |F9| Numeric, no float chars, w/sign*
=uNUMFn EQU    #FA      |FA| Numeric, w/float chars, no sign*
=uNUMFs EQU  (uNUMFn)+1 |FB| Numeric, w/float chars, w/sign*
=uNUMEn EQU    #FC      |FC| Numeric, w/Exponent, no sign*
=uNUMEs EQU  (uNUMEn)+1 |FD| Numeric, w/Exponent, w/sign*
*
*  *Note: these numeric delimiters have values that
*         determine the status bit setting in USING execute.
*********************************************************
```

```
                *
                *  The above token values fall in the following ranges.
                *  Values in parentheses are presently not used, and are
                *  reserved for future use:
                *
                *   Tokens not identifying the end of a field,
                *   not used in backward  search:
                *     (any value < D0) D0 D1 D2 D3 D4 (D5) (D6) (D7)
                *
                *   Tokens not identifying the end of a field,
                *   and used in backward search:
                *     D8 D9 DA DB DC (DD) DE DF E0 (E1) (E2) (E3) (E4) (E5)
                *
                *   Tokens identifying the end of a field (including those
                *   that start a new one), not used in backward search:
                *     (E6) (E7) (E8) (E9) (EA) (EB) (EC) (ED) EE EF F0
                *
                *   Tokens identifying the end of a field (including those
                *   that start a new one), used in backward search:
                *     F1 (F2) (F3) F4 (F5) F6 F7 F8 F9 FA FB FC FD (FE) (FF)
                *
                *  In truth, future applications may use ANY token values
                *  (or ASCII bytes, including any value above D0 if desired)
                *  for execution, providing that these token values:
                *      1) are "protected" from the execution and backward
                *         search routines by appropriate tokens (such as
                *         uIMXCH and uJMPst, respectively).  For instance,
                *         using the token value 44 (ASCII "D") for anything
                *         other than to output a digit will require a uIMXCH
                *         token to handle it.  Or including a 5-nibble pointer
                *         (which might contain, say, an F7 byte, by chance)
                *         requires a uIMXCH token to execute around it, and
                *         a uJMPst token to jump backwards over it.
                *
                *     or,
                *      2) cannot look like a token which will be executed
                *         or affect the backward search.  For instance,
                *         the MATH ROM places the two bytes uIMXCH and an
                *         ASCII ")" in the token stream; since an ASCII ")"
                *         won't cause any problems during execution or back-
                *         up, they are not protected.
                *
                *  Specifically, any token which may be found by the backward
                *  search routine is subject to the following rule:
                *      1) A value >= F1 identifies a delimiter in the IMinit
                *         routine.  IMinit will replace this token with the
                *         new one which identifies the type of field.
                *
                *  Any token which may be found by the execution routine is
                *  subject to the following two rules:
                *      1) A value >=F6 identifies the start of an output
                *         field.  EXPEXC will be called to evaluate the next
                *         output item, and formatting will be attempted.
                *      2) During the float-check and skip-checks (**), any
                *         token value >= E6 identifies the end of the check.
```

```
144             *       Symbols beyond this will not be counted; if you
145             *       want the count to continue beyond such a token,
146             *       you must protect it with a uMULT token, a uSTRPT
147             *       token, or a combination of uIMXCH with either.
148             *
149             *       (**) The float-check is performed on numeric "D"
150             *       fields to count the number of positions to float
151             *       editing symbols and signs.  The skip-check is
152             *       performed on any numeric field to count the number
153             *       of spaces before displaying NaN or Inf, or to count
154             *       the number of positions to fill with *'s in the
155             *       case of an IMAGE Ovfl warning.
156             *
157             *  See the pIMXCH poll documentation for examples of
158             *  protecting tokens from execution and backward searches.
159             *
160             ****************************************************************
```

```
                EJECT
*****************************************************************
*--- Status bits
*
*---- These status bits must be preserved during execution!
=sMULT  EQU    8              Multiplier pending.
=sSIGN  EQU    9              Sign specified.
=sFOUND EQU    10             Field found.
=sRDX   EQU    11             Radix found.
*
*---- These status bits can be changed during execution.
*    (status bits 0,1,2 are used for numeric flags in xqt)
=sXQT   EQU    0              Start executing.
=sC/P   EQU    1              C/P pending.
=sCntg  EQU    2              Counting digits.
=sInit  EQU    3              Field initialized.
*    (status bits 6,7 are used for complex output in xqt)
=InhEOL EQU    4              Same as SB&IO !!!   (Left =0)
=sSTOP  EQU    5              Stop backward search.
=sSpecl EQU    6              Special handling (used in xqtn)
=sCplxP EQU    7              Complex field pending.
*
```

```
183                 EJECT
184           ****************************************************************
185           *  Bits for character masks
186           *
187            X-chr  EQU    2^15           X: "blank"
188            D-chr  EQU    2^14           D: digit
189            A-chr  EQU    2^13           A: string char
190            Pt-chr EQU    2^12           Decimal point
191            Dblqt  EQU    2^11           Dbl quote: literal delim
192            Sglqt  EQU    2^10           Single quote: literal delim
193            S-chr  EQU    2^9            S: sign
194            M-chr  EQU    2^8            M: sign
195            Z-chr  EQU    2^7            Z: digit
196            E-chr  EQU    2^6            E: exponent
197            C-chr  EQU    2^5            C: separator
198            astrsk EQU    2^4            *: digit
199            Z1-chr EQU    2^3            Z: unit's digit A
200            P-chr  EQU    2^2            P: separator
201            R-chr  EQU    2^1            R: radix
202            Nf     EQU    2^0            Nxtfld flag: return to Nxtf15.
203            ed     EQU    (X-chr)+(Dblqt)+(Sglqt)   Edit chars
204            CP     EQU    (C-chr)+(P-chr)           Separators
205            SM     EQU    (S-chr)+(M-chr)           Sign chars
206            Rx     EQU    (Pt-chr)+(R-chr)          Radix chars
207            edSMRx EQU    (ed)+(SM)+(Rx)
208            CPE    EQU    (CP)+(E-chr)
209           *
```

```
210                 EJECT
211             ****************************************************************
212             **
213             **  IMAGE strings are not parsed until the string is accessed
214             **  for DISP USING execution (or PRINT USING, ENTER USING,
215             **  etc.).
216             **
217             **  Parsing of an IMAGE string is no simple task; the syntax is
218             **  very flexible, while there are more than two dozen
219             **  stringent rules.
220             **
221             **  There are two types of IMAGE strings.  For example,
222             **
223             **    1) DISP USING 100; X
224             **         where line 100 contains an IMAGE statement
225             **         with image string
226             **    2) DISP USING <string expression>; X
227             **         where <string expression> is a combination of
228             **         string constants or variables, such as
229             **           "3X,4Z,A" , or Y$(2) , or S$&"5X,"&FNI$
230             **
231             **  When a DISP USING statement is executed, the image string
232             **  is copied to the end of available memory, verbatim.  In
233             **  case 1), above, the image routines perform the copy; in
234             **  case 2), the routine EXPEXC (expression execute) evaluates
235             **  the expression and puts it at the end of available memory.
236             **  In either case, a ulMend byte is placed immediately
237             **  following the copy of the image string, so that the parse
238             **  routines can identify the end of the string.  Then this
239             **  copy is used to build a stream of tokens for execution of
240             **  the DISP USING statement.
241             **
242             **  A tokenized image string can only reside in RAM, since loop
243             **  counters (symbols with multipliers generate loop counters)
244             **  are stored as part of the tokenized stream.  (An
245             **  application could conceivably store tokenized image streams
246             **  in ROM if no multipliers were used.  It would of course
247             **  need a special symbol to reference the fixed tokens, and
248             **  poll handlers to position the image execution routines into
249             **  the token stream.)
250             **
251             **  In the discussion which follows, AvMemSt refers to
252             **  "Available Memory Start", the memory location whose address
253             **  is stored in AVMEMS.  Similarly, AvMemEnd refers to
254             **  "Available Memory End", the memory location whose address
255             **  is stored in AMVEME.  This discussion also refers
256             **  frequently to the IMAGE tokens which are equated at the
257             **  beginning of this module; see the list of equates for the
258             **  names and meanings of the tokens.
259             **
260             **  As an example of the first step in parsing an image string,
261             **  the following statement causes memory to be set up as
262             **  indicated:
263             **
264             **      DISP USING "3D.D,12A"; P,N$
```

```
265             **
266             **   0 <- AvMemSt+                    (image  AvMemEnd -> FFFFF
267             **          |                          copy)    |
268             **   -------+-----------------------+--------+-+------------
269             **          |          xqt tokens.. |3D.D,12A| |
270             **   -------+-----------------------+--------+-+------------
271             **                                             ^
272             **                                          uIMend token
273             **
274             **  The basic scheme behind image parsing can be described in
275             **  this algorithm:
276             **
277             **     1) parse the image string until a field is encountered
278             **        which requires an output item, or until end of image
279             **        string.
280             **     2) stop parsing, execute all pending fields
281             **     3) if end of string, goto (4), else goto 1).
282             **     4) if more items in output list, reposition execution
283             **        pointer to start of token stream and restart execu-
284             **        tion until output list exhausted.
285             **
286             **  It can be seen, then, that the IMAGE routines alternate
287             **  between parsing and executing.  There are several reasons
288             **  for this design; two important ones are 1) it does not
289             **  require the entire image string to be parsed if the output
290             **  list is short, and 2) it allows polls to be issued between
291             **  fields so that "external" execution can be performed on the
292             **  tokens (such as for ENTER USING).
293             **
294             **  The token stream is organized according to the image
295             **  fields.  Every time a new field is encountered, a
296             **  delimiting token is placed in the stream, and two 4-nibble
297             **  counters are initiated in the token stream to keep track of
298             **  the number of digits before and after a decimal point (if
299             **  it turns out to be a numeric field).  Then ASCII bytes are
300             **  placed in the token stream to direct the execution routines
301             **  as to the type of output to generate.  Periodically, the
302             **  token stream has to be scanned backwards to change a field
303             **  delimiter (such as when an E is encountered), or update the
304             **  digit counters (for instance, when the field ends).
305             **
306             **  The best way to describe the process is with an example.
307             **  Assume we are executing an IMAGE string as follows:
308             **
309             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
310             **
311             **  We will build the token stream here, byte by byte.  As the
312             **  parsing progresses, the current character in the image
313             **  string will be indicated with a "^", as follows:
314             **
315             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
316             **     ^
317             **     (i.e., we are about to parse the first character)
318             **
319             **  To keep the description byte-oriented, the image symbols
```

```
320             **  will be abbreviated as shown:
321             **     uA  = uALit    "A" literal field
322             **     uB  = uLOOPB   Loop on byte
323             **     uC  = uCPLXC   Complex field closed
324             **     uD  = uDELIM   Delimiter
325             **     uE  = uNUMEn   Numeric, w/Exponent, no sign
326             **     uF  = uNUMFn   Numeric, w/float chars, no sign
327             **     uG  = uNUMFs   Numeric, w/float chars, w/sign
328             **     uH  = uHKB^    H,K,B or ^ field
329             **     uI  = uIMsta   IMAGE string start
330             **     uJ  = uJMPst   Jump over string pointer (14 nibs)
331             **     uK  = uIMbck   Poll for backward search handler
332             **     uL  = uLOOPS   Loop on string (12 nibs)
333             **     uM  = uMULT    Multiplier
334             **     uN  = uNUMNn   Numeric, no float chars, no sign
335             **     uO  = uOPNWM   Open loop with multiplier
336             **     uP  = uLOOPP   Loop on parentheses (variable #bytes)
337             **     uQ  = uOPNM-   Open loop with mult, decremented
338             **     uR  = uRESTP   Restart parse
339             **     uS  = uSTRPT   String pointer
340             **     uT  = uNUMEs   Numeric, w/Exponent, w/sign
341             **     uU  = uNUMNs   Numeric, no float chars, w/sign
342             **     uW  = uOPNNM   Open loop without multiplier
343             **     uX  = uIMXCH   Strange execution character
344             **     u8  = uJMPdl   Jump over unfilled delimiter (8 nibs)
345             **     u9  = uJMP{}   Jump over paren loop ptr (9 nibs
346             **     u$  = uIMend   IMAGE string end
347             **
348             **  and the symbols
349             **     =#  =D  =.  =X  =A  =M  =Z  =E  =S  =K   and  =)
350             **  will refer to the ASCII byte values for the characters.
351             **
352             **  The example includes a complex field, which is handled by a
353             **  poll in the MATH ROM; this is shown to describe one
354             **  implementation of image token handling.
355             **
356             **  Here goes....
357             **
358             **  The image string has been copied into AvMem, verbatim, and
359             **  a uIMend token immediately follows it.  The parse pointer
360             **  points to the first character; tokens will be built below
361             **  this area.  A uIMsta (image start) token is immediately
362             **  written as the first token, and the execution anchor (the
363             **  location where execution will start) is set to this token:
364             **
365             **                                                        uI
366             **                                                       (1)
367             **
368             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
369             **     ^
370             **
371             **                                                     =# uI
372             **                                                    (2)(1)
373             **
374             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
```

```
375          **      ^
376          **
377          **                                          40 00 u0 00 uM =# uI
378          **                                         (7--6)(5--4)(3)(2)(1)
379          **
380          **  As in any multiplied field, (4) and (5) will become the
381          **  reference counter, and (6) and (7) will become the loop
382          **  counter.  NOTE: the bytes (6) and (7) show the multiplier
383          **  value "0004", written backwards as it would show up in
384          **  memory.  This convention is followed throughout the
385          **  example.
386          **
387          **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
388          **      ^
389          **
390          **                       u8  00  00 00 uD 40 00 u0 00 uM =# uI
391          **                      (12)(11)(10)(9)(8)(7--6)(5--4)(3)(2)(1)
392          **
393          **  The "(" symbol defines the first field.  Any time a new
394          **  field is found, tokens (8) through (12) are written out.
395          **
396          **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
397          **       ^
398          **
399          **         30  00  u0  00  uM  u8  00  00 00 uD 40 00 u0 00 uM
400          **       (17--16)(15--14)(13)(12)(11)(10)(9)(8)(7--6)(5--4)(3)
401          **
402          **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
403          **       ^
404          **
405          **  This is the first output symbol in the output field, so we
406          **  now can identify the first field with a uNUMFn (numeric,
407          **  with floating symbols, no sign).  A backward search finds
408          **  the delimiter token at (8); however, there are pending
409          **  fields to be executed (for the first field, this may not
410          **  really be true, but execution starts anyway).  Therefore
411          **  token (8) is replaced with a uRESTP and execution starts.
412          **                                              V
413          **         30  00  u0  00  uM  u8  00  00 00 uR 40 00 u0 00 uM
414          **       (17--16)(15--14)(13)(12)(11)(10)(9)(8)(7--6)(5--4)(3)
415          **
416          **  The execution routine writes several storage fields below
417          **  token (17), adjusts AvMemEnd to protect them, sends out a
418          **  pIMXQT poll, and then executes the pending tokens.  Token
419          **  (3) causes the loop counter in (7--6) to be decremented,
420          **  and replaces (5) with a uOPNM- token to indicate that the
421          **  loop counter has been decremented:
422          **                                                 V     V
423          **         30  00  u0  00  uM  u8  00  00 00 uR 30 00 uQ 00 uM
424          **       (17--16)(15--14)(13)(12)(11)(10)(9)(8)(7--6)(5--4)(3)
425          **
426          **  In all, tokens (1) through (8) are executed; token (8)
427          **  causes the parse routines to be invoked again, starting at
428          **  the last token which was parsed, the "D".  The execution
429          **  anchor is set at token (8), the location where execution
```

```
430           **  will start again.
431           **
432           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
433           **        ^
434           **
435           **  Again, this is the first output character in an output
436           **  field, so we back up through the tokens and re-write (8)
437           **  with:
438           **                                                  V
439           **         30  00  u0  00  uM  u8  00  00 00 uF 30 00 uQ 00 uM
440           **       (17--16)(15--14)(13)(12)(11)(10)(9)(8)(7--6)(5--4)(3)
441           **
442           **  Simultaneously, the multiplier at (13) is identified to be
443           **  associated with the "D", so the reference counter is filled
444           **  in, and the "D" is written out with a token to define the
445           **  loop:
446           **                       V
447           **      uB  =D  30  00  30  00  uM  u8  00  00 00 uF 30 00 uQ
448           **     (19)(18)(17--16)(15--14)(13)(12)(11)(10)(9)(8)(7--6)(5-
449           **
450           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
451           **         ^
452           **
453           **  A lower case symbol is always converted to uppercase:
454           **
455           **       =D  uB  =D  30  00  30  00  uM  u8  00  00 00 uF 30
456           **      (20)(19)(18)(17--16)(15--14)(13)(12)(11)(10)(9)(8)(7--
457           **
458           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
459           **          ^
460           **
461           **  The radix symbol causes both 4-nibble counters (9) through
462           **  (12) to be filled in with the number of digits before the
463           **  radix.  Also, the symbol is written out:
464           **                                                 V       V
465           **       =.  =D  uB  =D  30  00  30  00  uM  40  00  40 00 uF
466           **     (21)(20)(19)(18)(17--16)(15--14)(13)(12)(11)(10)(9)(8)
467           **
468           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
469           **           ^
470           **
471           **  Only a "D" is allowed after a radix (not "Z" or "*"), but
472           **  in fact, a "D" after the radix acts like a "Z".  That is,
473           **  any leading zeroes in a fraction ARE displayed.  So this
474           **  symbol is written out as a "Z":
475           **
476           **       =Z  =.  =D  uB  =D  30  00  30  00  uM  40  00  40 00
477           **     (22)(21)(20)(19)(18)(17--16)(15--14)(13)(12)(11)(10)(9)
478           **
479           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
480           **            ^
481           **
482           **         20  00  u0  00  uM  =Z  =.  =D  uB  =D  30  00  30
483           **       (27--26)(25--24)(23)(22)(21)(20)(19)(18)(17--16)(15--
484           **
```

```
485           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
486           **            ^
487           **
488           **  A delimiter means that the previous field is terminated;
489           **  since it is a numeric field, the total number of digits
490           **  must be entered in its field delimiter.  A backward search
491           **  is initiated that finds its delimiter at (8), and the total
492           **  number of digits is filled in:
493           **                                          V
494           **       =.  =D  uB  =D  30  00  30  00  uM  50  00  40 00 uF
495           **     (21)(20)(19)(18)(17--16)(15--14)(13)(12)(11)(10)(9)(8)
496           **
497           **  Then a new field is opened:
498           **
499           **         u8  00  00  00  uD  20  00  u0  00  uM  =Z  =.  =D
500           **       (32)(31)(30)(29)(28)(27--26)(25--24)(23)(22)(21)(20)
501           **
502           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
503           **              ^
504           **
505           **         30  00  u0  00  uM  u8  00  00  00  uD  20  00  u0
506           **       (37--36)(35--34)(33)(32)(31)(30)(29)(28)(27--26)(25-
507           **
508           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
509           **              ^
510           **
511           **  When an imbedded literal is found, we write out a uSTRPT
512           **  token and a 5-nibble offset which points to the address of
513           **  the literal in the copied image string.  This 5-nibble
514           **  offset is shown as "<ofs>".
515           **
516           **       <ofs> uS  30  00  u0  00  uM  u8  00  00  00  uD  20
517           **       (39)(38)(37--36)(35--34)(33)(32)(31)(30)(29)(28)(27-
518           **
519           **  Now the parse routines scan the image string until the
520           **  matching quote is found, counting the length of the
521           **  imbedded literal (in bytes).  This length is stored as a
522           **  5-nibble field shown as "<len>" (in this case, the length
523           **  is 3, and would appear in RAM as "30000").
524           **
525           **        uJ<len><ofs> uS  30  00  u0  00  uM  u8  00  00  00
526           **       (41)(40)(39)(38)(37--36)(35--34)(33)(32)(31)(30)(29)
527           **
528           **  Simultaneously, the multiplier at (33) is identified to be
529           **  associated with the imbedded literal.  A uLOOPS token is
530           **  written out to define the loop, and the reference counter
531           **  is filled in.
532           **                                    V
533           **       uL uJ<len><ofs> uS  30  00  30  00  uM  u8  00  00  00
534           **     (42)(41)(40)(39)(38)(37--36)(35--34)(33)(32)(31)(30)(29)
535           **
536           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
537           **                    ^
538           **
539           **  The comma is a new delimiter:
```

```
540             **
541             **      u8  00  00  00  uD  uL uJ<len><ofs> uS  30  00  30  00
542             **     (47)(46)(45)(44)(43)(42)(41)(40)(39)(38)(37--36)(35--34)
543             **
544             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
545             **                      ^
546             **
547             **  The left parenthesis is also a new delimiter.  But a check
548             **  is made to see if a delimiter has just been opened, in
549             **  order to save RAM.  Since this is the case, the BldIMG
550             **  pointer is backed up 5 bytes, to (42), so that the new
551             **  field will overwrite the one already opened.
552             **
553             **                              uL uJ<len><ofs> uS  30  00  30  00
554             **                             (42)(41)(40)(39)(38)(37--36)(35--34)
555             **
556             **  Before opening a new field, a left parenthesis without a
557             **  multiplier has to be identified so that the closing
558             **  parenthesis can be matched later:
559             **
560             **      u8  00  00  00  uD  uW  uL uJ<len><ofs> uS  30  00  30
561             **    (48)(47)(46)(45)(44)(43)(42)(41)(40)(39)(38)(37--36)(35-
562             **
563             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
564             **                      ^
565             **
566             **  The entire multiplier is scanned, to fill in the multiplier
567             **  counter (remember the convention of writing multipliers
568             **  backwards):
569             **
570             **      32  70  uD  00  uM  u8  00  00  00  uD  uW  uL uJ<len>
571             **    (53--52)(51--50)(49)(48)(47)(46)(45)(44)(43)(42)(41)(40)
572             **
573             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
574             **                        ^
575             **
576             **  The "X" also closes the multiplier at (49).
577             **                      V   V
578             **      uB  =X  32  70  32  70  uM  u8  00  00  00  uD  uW  uL
579             **     (55)(54)(53--52)(51--50)(49)(48)(47)(46)(45)(44)(43)(42)
580             **
581             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
582             **                        ^
583             **
584             **  A closing parenthesis initiates a backward search that
585             **  scans for the nearest open left parenthesis; in this case,
586             **  it is at token (43).  Since this parentheses field does not
587             **  have a multiplier, (43) is simply zeroed to "hide" it from
588             **  further backward searches.  In addition, a right
589             **  parenthesis is a new delimiter.
590             **                                                      V
591             **      uB  =X  32  70  32  70  uM  u8  00  00  00  uD  00  uL
592             **     (55)(54)(53--52)(51--50)(49)(48)(47)(46)(45)(44)(43)(42)
593             **
594             **      u8  00  00  00  uD  uB  =X  32  70  32  70  uM  u8  00
```

```
595          **     (60)(59)(58)(57)(56)(55)(54)(53--52)(51--50)(49)(48)(47)
596          **
597          **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
598          **                              ^
599          **
600          **  The comma is a new delimiter, but, again, it is detected
601          **  that a delimiter has just been opened; in order to save
602          **  some RAM, the BldIMG pointer is backed up to (55) so that
603          **  the 5 bytes simply overwrite the present ones:
604          **       V   V   V   V   V
605          **       u8  00  00  00  uD  uB  =X  32  70  32  70  uM  u8  00
606          **     (60)(59)(58)(57)(56)(55)(54)(53--52)(51--50)(49)(48)(47)
607          **
608          **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
609          **                              ^
610          **
611          **  Write out length and offset to imbedded literal "$":
612          **
613          **      uJ<len><ofs> uS  u8  00  00  00  uD  uB  =X  32  70  32
614          **     (64)(63)(62)(61)(60)(59)(58)(57)(56)(55)(54)(53--52)(51-
615          **
616          **
617          **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
618          **                                 ^
619          **
620          **       40  00  u0  00  uM uJ<len><ofs> uS  u8  00  00  00  uD
621          **     (69--68)(67--66)(65)(64)(63)(62)(61)(60)(59)(58)(57)(56)
622          **
623          **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
624          **                                 ^
625          **
626          **  Finally, we find the first output symbol for an output
627          **  field.  This causes a backward search to the pending
628          **  delimiter at (56).  Since there are pending output fields
629          **  which need to be executed, (56) is overwritten with a
630          **  uRESTP token, and execution begins:
631          **                                                        V
632          **       40  00  u0  00  uM uJ<len><ofs> uS  u8  00  00  00  uR
633          **     (69--68)(67--66)(65)(64)(63)(62)(61)(60)(59)(58)(57)(56)
634          **
635          **  The execution routine writes several storage fields below
636          **  token (69), adjusts AvMemEnd to protect them, sends out a
637          **  pIMXQT poll, and then executes the pending tokens.  Tokens
638          **  (8) through (56) are executed.  Token (23) causes the loop
639          **  counter in (27--26) to be decremented, and replaces (25)
640          **  with a uOPNM- token to indicate that the loop counter has
641          **  been decremented:
642          **                              V       V
643          **         u8  00  00  00  uD  10  00  uQ  00  uM  =Z  =.  =D
644          **       (32)(31)(30)(29)(28)(27--26)(25--24)(23)(22)(21)(20)
645          **
646          **  Token (56) causes the parse routines to be invoked again,
647          **  starting at the last token which was parsed, the "D".  The
648          **  execution anchor is set at token (56), the location where
649          **  execution will start again.
```

```
650             **
651             **      #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
652             **                                    ^
653             **
654             **  Again, this is the first output symbol in an ouput field,
655             **  so a backward search finds (56) to define the field:
656             **                                                          V
657             **        40  00  u0  00  uM uJ<len><ofs> uS  u8  00  00  00  uF
658             **     (69--68)(67--66)(65)(64)(63)(62)(61)(60)(59)(58)(57)(56)
659             **
660             **  Simultaneously, the multiplier at (65) is closed, and the
661             **  parsed symbol is written out:
662             **                        V
663             **        uB  =D  40  00  40  00  uM uJ<len><ofs> uS  u8  00  00
664             **     (71)(70)(69--68)(67--66)(65)(64)(63)(62)(61)(60)(59)(58)
665             **
666             **      #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
667             **                                    ^
668             **
669             **        60  00  u0  00  uM  uB  =D  40  00  40  00  uM uJ<len>
670             **     (76--75)(74--73)(72)(71)(70)(69--68)(67--66)(65)(64)(63)
671             **
672             **      #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
673             **                                   ^
674             **
675             **  Write out offset and length for "?" literal, close
676             **  multiplier at (72):
677             **                                 V
678             **        uL uJ<len><ofs> uS  60  00  60  00  uM  uB  =D  40  00
679             **     (81)(80)(79)(78)(77)(76--75)(74--73)(72)(71)(70)(69--68)
680             **
681             **      #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
682             **                                    ^
683             **
684             **        =D  uL uJ<len><ofs> uS  60  00  60  00  uM  uB  =D  40
685             **     (82)(81)(80)(79)(78)(77)(76--75)(74--73)(72)(71)(70)(69-
686             **
687             **      #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
688             **                                     ^
689             **
690             **  The sign symbol changes the type of field to uNUMFs
691             **  (numeric with floating symbols, with sign).  A backward
692             **  search changes the delimiter token at (56):
693             **                                                          V
694             **        40  00  40  00  uM uJ<len><ofs> uS  u8  00  00  00  uG
695             **     (69--68)(67--66)(65)(64)(63)(62)(61)(60)(59)(58)(57)(56)
696             **
697             **  And the "S" is placed in the token stream:
698             **
699             **        =S  =D  uL uJ<len><ofs> uS  60  00  60  00  uM  uB  =D
700             **     (83)(82)(81)(80)(79)(78)(77)(76--75)(74--73)(72)(71)(70)
701             **
702             **
703             **      #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
704             **                                     ^
```

```
705             **
706             **  This comma closes the numeric field.  A backward search is
707             **  initiated to find the delimiter at (56) to fill in the
708             **  total number of digits:
709             **                                                V       V
710             **       40  00  40  00  uM uJ<len><ofs> uS  50  00  50  00  uG
711             **     (69--68)(67--66)(65)(64)(63)(62)(61)(60)(59)(58)(57)(56)
712             **
713             **  The delimiter also opens a new field:
714             **
715             **       u8  00  00  00  uD  =S  =D  uL uJ<len><ofs> uS  60  00
716             **     (88)(87)(86)(85)(84)(83)(82)(81)(80)(79)(78)(77)(76--75)
717             **
718             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
719             **                                       ^
720             **
721             **       30  00  u0  00  uM  u8  00  00  00  uD  =S  =D  uL uJ
722             **     (93--92)(91--90)(89)(88)(87)(86)(85)(84)(83)(82)(81)(80)
723             **
724             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
725             **                                        ^
726             **
727             **  The "A" is the first output symbol in an output field, so a
728             **  backward search is initiated to find the pending delimiter
729             **  at (84).  Since there are pending fields to execute, (84)
730             **  is replaced with a uRESPT token, and execution begins:
731             **                                             V
732             **       30  00  u0  00  uM  u8  00  00  00  uR  =S  =D  uL uJ
733             **     (93--92)(91--90)(89)(88)(87)(86)(85)(84)(83)(82)(81)(80)
734             **
735             **  The execution routine writes several storage fields below
736             **  token (93), adjusts AvMemEnd to protect them, sends out a
737             **  pIMXQT poll, and then executes the pending tokens.  Tokens
738             **  (56) through (84) are executed; token (84) causes the parse
739             **  routines to be invoked again, starting at the last token
740             **  which was parsed, the "A".  The execution anchor is set at
741             **  token (84), the location where execution will start again.
742             **
743             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
744             **                                        ^
745             **
746             **  As the first output symbol in an output field, the "A"
747             **  causes a backward search to fill in the delimiter token at
748             **  (84):
749             **                                             V
750             **       30  00  u0  00  uM  u8  00  00  00  uA  =S  =D  uL uJ
751             **     (93--92)(91--90)(89)(88)(87)(86)(85)(84)(83)(82)(81)(80)
752             **
753             **  Simultaneously, the multiplier at (89) is closed, and the
754             **  parsed symbol is written out:
755             **                       V
756             **       uB  =A  30  00  30  00  uM  u8  00  00  00  uA  =S  =D
757             **     (95)(94)(93--92)(91--90)(89)(88)(87)(86)(85)(84)(83)(82)
758             **
759             **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
```

```
760           **                                        ^
761           **
762           **  The delimiter closes the current field, causing the symbol
763           **  counters at (88) and (86) to be filled (this is done for
764           **  string fields, as well as numeric fields; it doesn't hurt,
765           **  and might be handy for some applications).
766           **                                    V     V
767           **      uB  =A  30  00  30  00  uM  30  00  30  00  uA  =S  =D
768           **     (95)(94)(93--92)(91--90)(89)(88)(87)(86)(85)(84)(83)(82)
769           **
770           **  The comma also opens a new field:
771           **
772           **      u8  00  00  00  uD  uB  =A  30  00  30  00  uM  30  00
773           **    (100)(99)(98)(97)(96)(95)(94)(93--92)(91--90)(89)(88)(87)
774           **
775           **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
776           **                                          ^
777           **
778           **  This is an unrecognized symbol (cannot start a field).
779           **  First, the parse routine assumes a new delimiter has been
780           **  found; but in doing so, it finds that a delimiter has just
781           **  been opened anyway.  So the BldIMG pointer is moved back 5
782           **  bytes to (95), to save RAM.
783           **
784           **                  uB  =A  30  00  30  00  uM  30  00  30  00
785           **                 (95)(94)(93--92)(91--90)(89)(88)(87)(86)(85)
786           **
787           **  Now, any unrecognized symbol causes a pIMCHR poll.  The
788           **  MATH ROM, if present, will recognize "C(" as the first
789           **  symbol in an output field, and will insert the following
790           **  tokens:
791           **
792           **      uR  uC  00  uB  =A  30  00  30  00  uM  30  00  30  00
793           **     (98)(97)(96)(95)(94)(93--92)(91--90)(89)(88)(87)(86)(85)
794           **
795           **  Token (96) is a flag that says (in this case=00) "the
796           **  complex field does not have a multiplier".  If it did have
797           **  a multiplier, the value of this byte would be nonzero
798           **  (=01).
799           **
800           **  The MATH ROM causes IMAGE execution to start on the pending
801           **  fields.
802           **
803           **  The execution routine writes several storage fields below
804           **  token (98), adjusts AvMemEnd to protect them, sends out a
805           **  pIMXQT poll, and then executes the pending tokens.  Tokens
806           **  (84) through (98) are executed; token (98) causes the parse
807           **  routines to be invoked again, starting at the last token
808           **  which was parsed, the "C".  The execution anchor is set at
809           **  token (98), the location where execution will start again.
810           **
811           **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
812           **                                          ^
813           **
814           **  Again, the unrecognized character causes a pIMCHR poll.
```

```
815          **  The MATH ROM, this time, seeing that it has already
816          **  intercepted this poll once (tokens (96)-(98) wouldn't be
817          **  there otherwise), changes the tokens as follows:
818          **       V
819          **       uK  uC  00  uB  =A  30  00  30  00  uM  30  00  30  00
820          **     (98)(97)(96)(95)(94)(93--92)(91--90)(89)(88)(87)(86)(85)
821          **
822          **  Token (98) will cause a poll during backward search when
823          **  the closing complex parenthesis is found.  After handling
824          **  this pIMCHR poll, the MATH ROM causes the image parser to
825          **  open a new field, and keep parsing as usual.
826          **
827          **       u8   00   00  00   uD  uK  uC  00  uB  =A  30  00  30
828          **     (103)(102)(101)(100)(99)(98)(97)(96)(95)(94)(93--92)(91-
829          **
830          **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
831          **                                              ^
832          **
833          **       20   00   u0   00   uM   u8   00   00   00  uD  uK
834          **     (108--107)(106--105)(104)(103)(102)(101)(100)(99)(98)
835          **
836          **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
837          **                                                 ^
838          **
839          **       u8   00   00   00   uD   20   00   u0   00   uM   u8
840          **     (113)(112)(111)(110)(109)(108--107)(106--105)(104)(103)
841          **
842          **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
843          **                                                    ^
844          **
845          **  This is a leading sign symbol, and does not cause the field
846          **  delimiter to be adjusted (only a "D", "Z" or "*" can do
847          **  that).
848          **
849          **       =M   u8   00   00   00   uD   20   00   u0   00   uM
850          **     (114)(113)(112)(111)(110)(109)(108--107)(106--105)(104)
851          **
852          **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
853          **                                                    ^
854          **
855          **  This is the first output character in an output field.  It
856          **  initiates a backward search for the field delimiter at
857          **  (109).  Initializing this field causes a pIMcp1 poll (since
858          **  the MATH ROM had set S7=1 earlier; it wants to know when a
859          **  numeric field is initialized inside a complex field).  The
860          **  MATH ROM checks to make sure that no more than 2 numeric
861          **  fields have been specified, and also clears the status bit
862          **  (S3=0) to inhibit execution of the pending fields; the
863          **  complex field must be completely closed before this is
864          **  allowed.  So the backward search merely re-writes token
865          **  (109), and continues parsing:
866          **                                        V
867          **       =Z   =M   u8   00   00   00   uU   20   00   u0   00
868          **     (115)(114)(113)(112)(111)(110)(109)(108--107)(106--105)
869          **
```

```
870           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
871           **                                                  ^
872           **
873           **  The radix, initiating a backward search, causes the
874           **  counter fields to be filled in with the number of digits
875           **  before the radix:
876           **                          V         V
877           **       =.   =Z   =M   10   00   10   00   uU   20   00   u0
878           **     (116)(115)(114)(113)(112)(111)(110)(109)(108--107)(106-
879           **
880           **
881           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
882           **                                                  ^
883           **
884           **  Remember, a "D" after a radix behaves like a "Z":
885           **
886           **       =Z   =.   =Z   =M   10   00   10   00   uU   20   00
887           **     (117)(116)(115)(114)(113)(112)(111)(110)(109)(108--107)
888           **
889           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
890           **                                                  ^
891           **
892           **  The "E" signals the end of digit symbols, so it initiates a
893           **  backward search to fill in the total number of digits in
894           **  the field.  It also changes the type of field to uNUMEs
895           **  (numeric, with exponent, with sign):
896           **                             V                   V
897           **       =Z   =.   =Z   =M   20   00   10   00   uT   20   00
898           **     (117)(116)(115)(114)(113)(112)(111)(110)(109)(108--107)
899           **
900           **  And an "E" causes digit output as if the symbols "ESZZZ"
901           **  were being executed:
902           **
903           **       =Z   =Z   =Z   =S   =E   =Z   =.   =Z   =M   20   00
904           **     (122)(121)(120)(119)(118)(117)(116)(115)(114)(113)(112)
905           **
906           **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
907           **                                                  ^
908           **
909           **  The closing parenthesis causes a backward search to find
910           **  the nearest open parenthesis, which is at (106).  (106) is
911           **  closed by copying the loop counter at (108--107).  A
912           **  5-nibble offset to the location of the left parenthesis,
913           **  shown as "<ptr>", is written to the token stream.
914           **                                                     V
915           **       =Z   =M   20   00   10   00   uT   20   00   20   00
916           **     (115)(114)(113)(112)(111)(110)(109)(108--107)(106--105)
917           **
918           **       u9<ptr>   uP   =Z   =Z   =Z   =S   =E   =Z   =.   =Z
919           **     (125)(124)(123)(122)(121)(120)(119)(118)(117)(116)(115)
920           **
921           **  The right parenthesis is also a delimiter, so a new field
922           **  is opened:
923           **
924           **       u8   00   00   00   uD   u9<ptr>   uP   =Z   =Z   =Z
```

```
925             **    (130)(129)(128)(127)(126)(125)(124)(123)(122)(121)(120)
926             **
927             **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
928             **                                                      ^
929             **
930             **  Since no new characters are written out here, the image
931             **  parser assumes a new field is to open.  But it checks and
932             **  finds that a new field has already been opened; to save RAM
933             **  it backs up the BldIMG pointer 5 bytes to (125).
934             **
935             **                                  u9<ptr>  uP  =Z  =Z  =Z
936             **                               (125)(124)(123)(122)(121)(120)
937             **
938             **  This new right parenthesis is to close the complex field.
939             **  The backward search will find the uIMbck token at (98), and
940             **  poll (pIMbck).  The MATH ROM verifies that 2, and only 2,
941             **  numeric fields were included, checks if the complex field
942             **  has a multiplier (if so, it writes out fields similar to
943             **  (123)-(125)), and changes the tokens as follows:
944             **                                                        V
945             **      00   20   00   uM   u8   00   00   00   uD  uX  uC  00
946             **   -107)(106--105)(104)(103)(102)(101)(100)(99)(98)(97)(96)
947             **
948             **  and
949             **       uX   =)  u9<ptr>  uP   =Z   =Z   =Z   =S   =E   =Z
950             **    (127)(126)(125)(124)(123)(122)(121)(120)(119)(118)(117)
951             **
952             **  The uIMXCH token at (98) will cause the MATH ROM to display
953             **  a "(" for the complex number, to evaluate the complex
954             **  expression and prepare it for formatting.  The uIMXCH token
955             **  at (127) is used simply to display a ")" to close the
956             **  complex field.
957             **
958             **    #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
959             **                                                      ^
960             **
961             **  This closing parenthesis matches the open one at token
962             **  (25).  A backward search finds the uOPNM- token, which
963             **  causes two actions: 1) the reference counter at (25) is
964             **  filled in by copying the loop counter at (27--26) and
965             **  adding 1 (since the loop counter has been marked as being
966             **  already decremented), and 2) an offset pointer is entered
967             **  in the token stream to point back to the beginning of this
968             **  parentheses loop.
969             **                                      V
970             **        u8  00  00  00  uD  10  00  20  00  uM  =Z  =.  =D
971             **       (32)(31)(30)(29)(28)(27--26)(25--24)(23)(22)(21)(20)
972             **
973             **       u9<ptr>  uP  uX  =)  u9<ptr>  uP  =Z  =Z  =Z
974             **    (130)(129)(128)(127)(126)(125)(124)(123)(122)(121)(120)
975             **
976             **  In addition, the parenthesis is a delimiter, so a new field
977             **  is opened:
978             **
979             **       u8   00   00   00   uD   u9<ptr>   uP   uX   =)   u9
```

```
980           **     (135)(134)(133)(132)(131)(130)(129)(128)(127)(126)(125)
981           **
982           **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
983           **                                                        ^
984           **
985           **  The "K" is the first output symbol in an output field, so a
986           **  backward search is initiated to find the pending delimiter
987           **  at (131).  Since there are pending fields to execute, (131)
988           **  is replaced with a uRESPT token, and execution begins:
989           **                                V
990           **        u8   00   00   00   uR   u9<ptr>   uP   uX   =)   u9
991           **     (135)(134)(133)(132)(131)(130)(129)(128)(127)(126)(125)
992           **
993           **  The execution routine writes several storage fields below
994           **  token (135), adjusts AvMemEnd to protect them, sends out a
995           **  pIMXQT poll, and then executes the pending tokens.  Tokens
996           **  (98) through (135) are executed; token (135) causes the
997           **  parse routines to be invoked again, starting at the last
998           **  token which was parsed, the "K".  The execution anchor is
999           **  set at token (135), the location where execution will start
1000          **  again.
1001          **
1002          **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
1003          **                                                        ^
1004          **
1005          **  Again, this is the first output symbol in an output field,
1006          **  so a backward search finds token (131) and replaces it with
1007          **  uHKB^.  The parsed symbol is also written out.
1008          **                                  V
1009          **        =K   u8   00   00   00   uH   u9<ptr>   uP   uX   =)
1010          **     (136)(135)(134)(133)(132)(131)(130)(129)(128)(127)(126)
1011          **
1012          **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
1013          **                                                        ^
1014          **
1015          **  This closing parenthesis matches the open one at token (5).
1016          **  A backward search finds the uOPNM- token, which causes two
1017          **  actions: 1) the reference counter at (5) is filled in by
1018          **  copying the loop counter at (7--6) and adding 1 (since the
1019          **  loop counter has been marked as being already decremented),
1020          **  and 2) an offset pointer is entered in the token stream to
1021          **  point back to the beginning of this parentheses loop.
1022          **                                                        V
1023          **         30  00  30  00  uM  40  00  40 00 uF 30 00 40 00 uM
1024          **       (17--16)(15--14)(13)(12)(11)(10)(9)(8)(7--6)(5--4)(3)
1025          **
1026          **        u9<ptr>   uP   =K   u8   00   00   00   uH   u9<ptr>
1027          **     (139)(138)(137)(136)(135)(134)(133)(132)(131)(130)(129)
1028          **
1029          **  In addition, the parenthesis is a delimiter, so a new field
1030          **  is opened:
1031          **
1032          **        u8   00   00   00   uD   u9<ptr>   uP   =K   u8   00
1033          **     (144)(143)(142)(1419(140)(139)(138)(137)(136)(135)(134)
1034          **
```

```
1035          **     #4(3Dd.D2(3"Ha!",(723x),"$"4D6"?"Ds,3A,C(2(MZ.dE)))K)
1036          **                                                          ^
1037          **
1038          **  The parse pointer now points to the uIMend byte (placed
1039          **  there before parsing began).  Since no new symbols are
1040          **  written out, the parser assumes a new field is about to
1041          **  start.  But it checks and finds that a new field was just
1042          **  opened, so the BldIMG pointer is backed up 5 bytes to
1043          **  (139).
1044          **                                    u9<ptr>   uP   =K   u8   00
1045          **                                (139)(138)(137)(136)(135)(134)
1046          **
1047          **  Parsing the uIMend byte simply causes the uIMend token to
1048          **  be placed in the BldIMG stream.  It also initiates a
1049          **  backward search, looking for an open parenthesis.  In this
1050          **  case, there are none; finding the uIMsta token at (1)
1051          **  signals that there are no unmatched parentheses.
1052          **
1053          **                                 u$   u9<ptr>   uP   =K   u8   00
1054          **                             (140)(139)(138)(137)(136)(135)(134)
1055          **
1056          **  Execution then begins as usual.  The execution routine
1057          **  writes several storage fields below token (140), adjusts
1058          **  AvMemEnd to protect them, sends out a pIMXQT poll, and then
1059          **  executes the pending tokens.  Tokens (135) through (140)
1060          **  are executed; token (140) causes the execution routines to
1061          **  test for the end-of-output list.
1062          **
1063          **  If at end-of-output, the statement is over; before exiting
1064          **  to NXTSTM the BldIMG pointer is positioned to the second
1065          **  token in the stream to test for an ASCII "#".  The presence
1066          **  of this symbol suppresses a CR-LF.
1067          **
1068          **  If there are more output items, the BldIMG stream is
1069          **  recycled (the execution anchor is moved back to token (1)),
1070          **  and the tokens are executed for the remaining output items.
1071          **  The parse routines will not be invoked again, since no
1072          **  uRESTP token will be encountered.
1073          **
1074          ****************************************************************
```

```
1075                    EJECT
1076            ********************************************************************
1077            ********************************************************************
1078            **
1079            ** Name:(S) pIMCHR  -  Poll for unrecognized IMAGE char
1080            **
1081            ** Category:   POLL
1082            **
1083            ** Type:       FPOLL
1084            **
1085            ** Purpose:
1086            **      To alert LEX files that an unrecognized character was
1087            **      found while parsing an IMAGE string.  If a LEX file
1088            **      doesn't handle it, "Invalid IMAGE" error will result.
1089            **
1090            ** Should poll be "Handled" (return with XM=0)?:
1091            **      Yes.
1092            **
1093            ** Meaning of "Handling" Poll (what does code do if handled?):
1094            **      Unrecognized character was accepted, IMAGE token
1095            **      stream was adjusted (if necessary) to process
1096            **      the character at execution.
1097            **
1098            ** Entry conditions for handler (registers, ST, RAM, etc.):
1099            **      Carry set (fast poll)
1100            **      B[A] = Poll number.
1101            **      HEX mode.
1102            **      P=0.
1103            **      R0(A)=points to current position in BldIMG token
1104            **         stream.  If any tokens are to be appended to
1105            **         the stream, they should be added below this
1106            **         point. Pointer goes to D1, usually.
1107            **      R0(9-5)=execution pointer. Next time execution of
1108            **         an IMAGE field starts, it will start here.
1109            **      R1(A)=address of unrecognized character which
1110            **         caused the poll. Pointer goes to D0, usually.
1111            **      R1(9-5)=length of IMAGE string (in nibbles)
1112            **      R1(S)=counter for complex numeric fields
1113            **      R2(A)=counter for digits in numeric field (also
1114            **         for A's in a literal field, but this counter
1115            **         is not used).
1116            **      R3(A)=Program Counter (D0 at entry of USING routine)
1117            **      R3(9-5)=address of start of IMAGE string.
1118            **      See USING routine header for explanation of status
1119            **         bits.
1120            **
1121            ** Normal exit conditions from handler if handled (ST, RAM,
1122            ** registers, etc.):
1123            **      HEX mode.
1124            **      XM=0.
1125            **      D1=points to current positon in BldIMG token
1126            **         stream.  If tokens have been added, D1 must
1127            **         have been moved; if not, D1 must have been
1128            **         set to the address in R0(A).
1129            **      D(A)= AvMemSt
```

```
1130            **      R1(A)=address of next parse character in IMAGE string.
1131            **         This pointer should be moved past the character
1132            **         which caused the poll.
1133            **      P is used to jump (see NOTE)
1134            **         P = 0: jump to Nxtfld
1135            **             1: jump to CkDlm+ (S3 must=1)
1136            **             2: jump to IMGxq1
1137            **      Other fields in R registers should be untouched,
1138            **         unless the poll handler has specific reasons
1139            **         to change them.
1140            **      Status bits should be untouched, unless the poll
1141            **         handler has specific reasons to change them.
1142            **
1143            ** Normal exit conditions from handler if not handled (ST, RAM,
1144            ** registers, etc.):
1145            **      Carry clear (POLL only).
1146            **      HEX mode.
1147            **      XM=1.
1148            **      R registers untouched.
1149            **      (If not handled by any LEX file, IMAGE routines issue
1150            **        an "Invalid IMAGE" error.)
1151            **
1152            ** Available subroutine levels:
1153            **      5
1154            **
1155            ** NOTE:
1156            **      IMAGE parsing and execution are very involved.
1157            **      Study the USING routine header and pIMbck, pIMcpi,
1158            **      pIMXCH and pIMXQT poll documentation to learn
1159            **      more about the process.  The USING routine header
1160            **      describes the meaning and values of the IMAGE
1161            **      tokens.
1162            **
1163            **      The IMAGE string and BldIMG token stream is kept
1164            **      in available memory, below AvMemEnd.  The BldIMG
1165            **      token stream is built backwards (toward address 0)
1166            **      from the boundary of the IMAGE string:
1167            **
1168            **            BldIMG tokens   IMAGE string   AvMemEnd
1169            **        ----------------------------------------------
1170            **            |               |    ?          |
1171            **        ----------------------------------------------
1172            **            ^                    ^ \
1173            **            |                    | unrecognized char
1174            **         pointer in R0(A)   pointer in R1(A)
1175            **
1176            **      Just because a character was encountered that your
1177            **      LEX file will accept, don't accept it blindly.
1178            **      For instance, don't accept a digit specifier in a
1179            **      literal field.  For cases like this, you have to
1180            **      back up through the BldIMG tokens to the field
1181            **      delimiter to see what type of field is being
1182            **      processed.  If the syntax of the new character
1183            **      doesn't meet your requirements, let the poll go
1184            **      on with XM=1 ("not handled").
```

```
1185          **
1186          **      Any strange characters put into the BldIMG stream
1187          **      by a poll handler which require special processing
1188          **      should be preceded by a uIMXCH token to alert the
1189          **      IMAGE execution routines that the poll handler will
1190          **      execute it.  Similarly, any strange characters
1191          **      which will adversely affect the backward searching
1192          **      during parse should be "protected" by a uJMPst token
1193          **      (which jumps over 14 nibbles), or a uIMbck token
1194          **      (which causes a pIMbck poll so that the poll handler
1195          **      can do the backup).  Backward searching during parse
1196          **      is performed for two reasons:
1197          **         1) to search for an open parenthesis (either to
1198          **            match a closing paren, or at the end of the
1199          **            IMAGE string to verify no unmatched parens).
1200          **         2) to search for delimiter (to initialize an out-
1201          **            put field, or to fill in the number of digits
1202          **            in a numeric field).
1203          **
1204          **      See the pIMbck poll documentation for appropriate
1205          **      use of the uIMbck token.  See the pIMXCH poll docu-
1206          **      mentation for examples of "protecting" the tokens.
1207          **
1208          **      Upon return from the pIMCHR poll, the poll handler
1209          **      can select three locations to jump to:
1210          **         Nxtfld -- This routine intializes a new field,
1211          **             and will accept only the normal start-of-
1212          **             field characters (such as D,X,Z,A,S,etc.)
1213          **             If a normal start-of-field character is not
1214          **             found, another pIMCHR poll will be issued.
1215          **             Nxtfld should be used if the unrecognized
1216          **             character is, say, a new type of digit
1217          **             specifier, a new editing symbol, or a new
1218          **             delimiter.  However, if the new character
1219          **             initiates an output field, you should jump
1220          **             to IMGxq1.
1221          **         CkDlm+ -- This routine checks for editing
1222          **             characters, then accepts only a standard
1223          **             delimiter (comma,"/","@",etc.).  If a
1224          **             delimiter is not found, another pIMCHR poll
1225          **             will be issued.  In order to jump to CkDlm+,
1226          **             S3 must be set=1 !  CkDlm+ should be used
1227          **             if the unrecognized character, say, termin-
1228          **             ates a field, or describes an entire non-
1229          **             output field (such as a new symbol which
1230          **             sounds the beeper).
1231          **         IMGxq1 -- This routine executes all pending
1232          **             IMAGE fields.  This should be done any time
1233          **             a new output field is initiated.  If the
1234          **             unrecognized character initiates a new out-
1235          **             put field, a uRESTP (restart IMAGE parse)
1236          **             token should be written into the BldIMG
1237          **             stream, and execution begun by jumping to
1238          **             IMGxq1.  A good example is the complex
1239          **             field, which intercepts the pIMCHR poll
```

```
1240            **              and causes a jump to IMGxq1 upon return.
1241            **
1242            ** What registers/RAM may be used if handled?:
1243            **      A,B,C,D,D0,D1,P
1244            **      R registers only to adjust values for specific reasons
1245            **
1246            ** What registers/RAM may be used if not handled?:
1247            **      A,B,C, D[15-5], D0, D1, P
1248            **      Don't change AVMEME pointer, or write to available
1249            **        memory below AvMemEnd.
1250            **
1251            ** Special memory/pointer considerations (are pointers funny?):
1252            **      The IMAGE string is stored just below AvMemEnd.  The
1253            **      BldIMG token stream is stored below that.  All this
1254            **      resides in available memory, so it is volatile (in the
1255            **      sense that someone can inadvertantly write over it, if
1256            **      they aren't careful).
1257            **
1258            ** Envisioned application(s):
1259            **      Well....
1260            **        1) Complex IMAGE fields
1261            **        2) A symbol which causes a one-time parsing of
1262            **           the IMAGE string (and stores it in an I/O buffer)
1263            **           for subsequent execution.  This would be much
1264            **           faster than parsing it each time.
1265            **        3) Allowing the "%" symbol to generate digit output.
1266            **        4) Specifying the "!" symbol, say, to generate a
1267            **           beep during IMAGE execution.
1268            **        5) Using square brackets to allow multiple-character
1269            **           replication.  E.g., "5[3DC]" would be equivalent
1270            **           to "3DC3DC3DC3DC3DC"
1271            **        6) ... and so on ...
1272            **
1273            ** History:
1274            **
1275            **    Date     Programmer              Modification
1276            **  --------   ----------   ---------------------------------
1277            **  12/08/82   MB           Implemented, documented.
1278            **
1279            ********************************************************************
1280            ********************************************************************
```

```
1281                    EJECT
1282            ********************************************************************
1283            ********************************************************************
1284            **
1285            ** Name:(S) pIMbck  -  Backward search, IMAGE parse
1286            **
1287            ** Category:   POLL
1288            **
1289            ** Type:       FPOLL
1290            **
1291            ** Purpose:
1292            **      Allow LEX files to handle unknown tokens while
1293            **      performing backward search during IMAGE parse.
1294            **
1295            ** Should poll be "Handled" (return with XM=0)?:
1296            **      Yes.
1297            **
1298            ** Meaning of "Handling" Poll (what does code do if handled?):
1299            **      Backward search over unknown tokens was performed
1300            **      properly.  One of two actions was performed:
1301            **         1) unknown field was closed
1302            **         2) unknown field was verified to be closed
1303            **            during final parentheses match.
1304            **
1305            ** Entry conditions for handler (registers, ST, RAM, etc.):
1306            **      Carry set.
1307            **      B[A] = Poll number.
1308            **      HEX mode.
1309            **      P=0.
1310            **      R1(A)=address of symbol which caused backward
1311            **        search (either a right parenthesis, or the
1312            **        end-of-image).
1313            **      R2(A)=address (in BldIMG stream) of the uIMbck
1314            **        token which caused the poll.
1315            **
1316            **      R0(A)=current position in BldIMG token stream.
1317            **        Next token to be entered must be written below
1318            **        this address.
1319            **      R0(9-5)=address to start next IMAGE execution
1320            **      R1(9-5)=length of IMAGE string (#nibbles)
1321            **      R1(S)=counter for 2 complex numeric fields.
1322            **      R3(A)=Program Counter
1323            **      R3(9-5)=address of start of IMAGE string.
1324            **
1325            ** Normal exit conditions from handler if handled (ST, RAM,
1326            ** registers, etc.):
1327            **      HEX mode.
1328            **      XM=0.
1329            **      D(A)=AvMemSt
1330            **      D1=current position in BldIMG (taken from R0(A),
1331            **         adjusted if necessary)
1332            **      Other R register fields unchanged
1333            **      See NOTE below for changes to BldIMG stream.
1334            **
1335            ** Normal exit conditions from handler if not handled (ST, RAM,
```

```
1336            ** registers, etc.):
1337            **      HEX mode.
1338            **      XM=1.
1339            **      R registers untouched.
1340            **      BldIMG token stream untouched.
1341            **      If not handled by any LEX file, IMAGE routines will
1342            **        issued an "Invalid IMAGE" error.
1343            **
1344            ** Available subroutine levels:
1345            **      4
1346            **
1347            ** NOTE:
1348            **      The pIMbck poll is issued only when a uIMbck token
1349            **      is encountered during backward search in IMAGE parse.
1350            **      The only way a uIMbck token could have been entered
1351            **      into the token stream is for a LEX file to have in-
1352            **      serted it during a pIMCHR poll.
1353            **
1354            **      Backward searching during IMAGE parse is performed
1355            **      for two reasons:
1356            **        1) To search for an open parenthesis: either to
1357            **           match a closing parenthesis (S5=0), or at the
1358            **           end-of-image to verify no unmatched parentheses
1359            **           (S5=1).  Use S5 to distinguish the two cases.
1360            **        2) To search for a field delimiter: to initiatiate
1361            **           an output field, or to fill in the number of
1362            **           digits in a numeric field.
1363            **      The pIMbck poll is issued only for case number 1 !!
1364            **      (The uIMbck token is ignored during backward search
1365            **      for a delimiter.)
1366            **
1367            **      This poll can be used by any new IMAGE syntax which
1368            **      uses parentheses to enclose a field (such as complex
1369            **      fields), or by an application which needs to know
1370            **      when the end-of-image has occurred (whether to check
1371            **      its own tokenization, or whatever).
1372            **
1373            **      Once this poll is issued, the backward search term-
1374            **      inates -- if handled, parsing continues at the point
1375            **      where the backward search was caused; if not handled
1376            **      by any LEX file, "Invalid IMAGE" is reported.
1377            **      Typically, a LEX file would expect to handle this
1378            **      poll only once -- to close the pending field (such
1379            **      as to close a complex field), or, failing to close
1380            **      it, to trap the error ("field not closed", such as
1381            **      unmatched parentheses) at end-of-image.  When it is
1382            **      handled properly (i.e., when the pending field is
1383            **      closed), the uIMbck token should be overwritten with
1384            **      another token so that the pIMbck poll is not issued
1385            **      again.  For instance, the MATH ROM, when handling the
1386            **      pIMCHR poll for a complex field, inserts the following
1387            **      tokens in the BldIMG stream:
1388            **                  uI  uC  u?  ...(existing BldIMG tokens)
1389            **                 (3) (2) (1)
1390            **          where
```

```
1391            **              uI =uIMbck token
1392            **              uC =uCPLXC token
1393            **              u? =flag to indicate whether multiplied field.
1394            **              uX =uIMXCH token (below)
1395            **      Later, when the closing parenthesis is found to close
1396            **      the complex field, the backward search will poll at
1397            **      token (3).  The MATH ROM will overwrite this token
1398            **      with a uIMXCH:
1399            **                        uX  uC  u?  ...(existing BldIMG tokens)
1400            **                       (3) (2) (1)
1401            **      Since the complex field was properly closed, a pIMbck
1402            **      poll need not be issued again.  Note that if the
1403            **      closing parenthesis had not been found, the pIMbck
1404            **      token would still be there at end-of-image.  End-of-
1405            **      image also performs a backward search to detect un-
1406            **      matched parentheses; during this pIMbck poll, the MATH
1407            **      ROM would find S5=1, issuing an "Invalid IMAGE" error.
1408            **
1409            **      If an application handles the poll and wishes the
1410            **      backward search to continue, it should either perform
1411            **      its own backward search (see "BACK2(" routine), or
1412            **      "erase" its uIMbck token from the BldIMG stream and
1413            **      reposition D1 and D0 so that the backward search
1414            **      is performed once again by the IMAGE routines.  That
1415            **      is, subtract 2 from R1(A) (so that it will point to
1416            **      the symbol which causes the backward search), and
1417            **      restore D1 from R0(A) (the current position in the
1418            **      BldIMG stream).  Or if the poll handler wants to be
1419            **      polled more than once, it can, each time, move its
1420            **      uIMbck token out the search area (write it below the
1421            **      current BldIMG address), and reposition D1 and D0
1422            **      as above to regenerate the backward search.
1423            **
1424            ** What registers/RAM may be used if handled?:
1425            **      A,B,C,D,D0,D1,P,R0(A),R2,R4
1426            **      Other fields in R registers may be adjusted as
1427            **        necessary.
1428            **      BldIMG tokens may be adjusted as necessary.
1429            **
1430            ** What registers/RAM may be used if not handled?:
1431            **      A,B,C,D[15-5],D0,D1,P,R4
1432            **      Do not write below AvMemEnd (contains BldIMG tokens)
1433            **
1434            ** Special memory/pointer considerations (are pointers funny?):
1435            **      The BldIMG stream resides in AvMem, below AvMemEnd.
1436            **
1437            ** Envisioned application(s):
1438            **      1) The MATH ROM uses the pIMbck poll to close complex
1439            **         image fields.  At that time, it checks whether
1440            **         2 (and only 2) numeric fields were included, and
1441            **         whether the field had a multiplier.  It also
1442            **         generates a uIMXCH token to execute the complex
1443            **         field, and another one to output a right paren-
1444            **         thesis.
1445            **      2) Say a LEX file implements an IMAGE symbol "="
```

```
1446            **          which causes pre-parsing of the image string
1447            **          (storing the tokens in an I/O buffer).  The
1448            **          syntax might be that it must be the first char-
1449            **          acter in the image string (even before a "#").
1450            **          A pIMCHR poll would be issued for the "="; the
1451            **          poll handler would insert a uIMbck token as the
1452            **          first token in BldIMG stream.  When the poll
1453            **          handler intercepts the pIMbck poll with S5=1,
1454            **          it would know that the entire image string had
1455            **          been parsed, and was ready to store away.
1456            **
1457            ** History:
1458            **
1459            **    Date      Programmer              Modification
1460            **  --------   ----------   ------------------------------------
1461            **  12/08/82   MB           Implemented, documented.
1462            **
1463            ********************************************************************
1464            ********************************************************************
```

```
1465                    EJECT
1466            ********************************************************************
1467            ********************************************************************
1468            **
1469            ** Name:(S) pIMcpi  -  Initializing IMAGE field in complex
1470            **
1471            ** Category:   POLL
1472            **
1473            ** Type:       FPOLL
1474            **
1475            ** Purpose:
1476            **      Alert MATH ROM that a field is being initialized while
1477            **      a complex field is pending.  Alert other LEX files that
1478            **      an output field is about to be initialized.
1479            **
1480            ** Should poll be "Handled" (return with XM=0)?:
1481            **      Yes.
1482            **
1483            ** Meaning of "Handling" Poll (what does code do if handled?):
1484            **      New field was verified to be numeric; total number
1485            **      of numeric fields in the complex field does not yet
1486            **      exceed 2.
1487            **
1488            ** Entry conditions for handler (registers, ST, RAM, etc.):
1489            **      Carry set.
1490            **      B[A] = Poll number.
1491            **      HEX mode.
1492            **      P=0.
1493            **      R1(A)=address of character in image string which
1494            **        initialized field (an output character such
1495            **        as D,Z,*,A,K)
1496            **      R0(A)=current position in BldIMG token stream
1497            **      R2(B)=proposed initializing token (identifies
1498            **        type of field)
1499            **      R2(XS)=0 (flag for IMAGE routines; don't change)
1500            **      R1(S)=counter for 2 complex numeric fields
1501            **
1502            **      R0(9-5)=address to start next IMAGE execution
1503            **      R1(9-5)=length of IMAGE string (nibbles)
1504            **      R3(A)=Program Counter
1505            **      R3(9-5)=address of start of IMAGE string.
1506            **
1507            ** Normal exit conditions from handler if handled (ST, RAM,
1508            ** registers, etc.):
1509            **      HEX mode.
1510            **      P=0
1511            **      XM=0.
1512            **      R2(B)=symbol which caused initialization (must be
1513            **        in upper case; fetched from address in R1(A))
1514            **      B(X)=contents of R2(X) from entry to poll handler
1515            **      D1=current position in BldIMG stream (from R0(A))
1516            **      S4=0 ("do not execute yet")
1517            **      D(A)=AvMemSt
1518            **
1519            ** Normal exit conditions from handler if not handled (ST, RAM,
```

```
1520          ** registers, etc.):
1521          **      HEX mode.
1522          **      XM=1.
1523          **      R registers untouched.
1524          **
1525          ** Available subroutine levels:
1526          **      4
1527          **
1528          ** NOTE:
1529          **      This is a specialized poll for the MATH ROM to handle
1530          **      complex image fields.  With some creative coding, the
1531          **      pIMcpi poll can be used by other LEX files.
1532          **
1533          **      The pIMcpi poll is only issued if S7=1 ("complex field
1534          **      being parsed") during image parse; and only when a
1535          **      new output field is being initialized in the BldIMG
1536          **      token stream.
1537          **
1538          **      There are two classes of poll handlers for pIMcpi.
1539          **         1) MATH ROM -- used to process numeric fields
1540          **              in a complex field.
1541          **            In a previous pIMCHR poll (issued at the "C("
1542          **            symbol), the poll handler must have:
1543          **             a) set S7=1
1544          **             b) set R1(S)=2
1545          **
1546          **         2) Other LEX files desiring to detect the init-
1547          **              ialization of any field.
1548          **            In a previous pIMCHR poll (issued at the
1549          **            point a new unrecognized symbol was found),
1550          **            the poll handler must have:
1551          **             a) set S7=1
1552          **             b) set R1(S)=0 (the MATH ROM will still
1553          **                intercept the pIMcpi poll, but if R1(S)
1554          **                is=0, it will exit "not handled")
1555          **             c) S7 must be set=0 before execution of the
1556          **                IMAGE tokens begins. (S7=1 during execu-
1557          **                tion will always invoke the MATH ROM;
1558          **                see pIMcpw poll documentation.)
1559          **
1560          **      Note that the pIMcpi poll was designed as a special
1561          **      poll for the MATH ROM.  Its use by any other ROM
1562          **      will conflict with complex fields.  In particular,
1563          **      a new symbol can use this poll as long as it and
1564          **      complex fields are syntactically mutually exclusive.
1565          **        -- If S7 has been set=1 by another LEX file
1566          **           then the MATH ROM will not handle the pIMCHR
1567          **           poll for a subsequent "C(" symbol.  In other
1568          **           words, setting S7=1 will cause an "Invalid
1569          **           IMAGE" when a complex field is found.
1570          **        -- Any application handling this poll cannot
1571          **           allow its new symbol within a complex field,
1572          **           since the MATH ROM, if it intercepts the poll
1573          **           first, will try to process it.  The counter
1574          **           in R1(S) will cause a conflict.
```

```
1575            **
1576            **      (Notwithstanding the above rule, there is probably
1577            **      a way for a pIMcpi poll handler to manage the use
1578            **      of R1(S) to allow complex fields within its own
1579            **      new field.  See the MATH ROM code for complete
1580            **      details.)
1581            **
1582            ** What registers/RAM may be used if handled?:
1583            **      A,B,C,D,D0,D1,P,R4,S4,S7
1584            **      R registers may be adjusted as necessary
1585            **      Tokens in BldIMG stream adjusted as necessary
1586            **
1587            ** What registers/RAM may be used if not handled?:
1588            **      A,B,C,D[15-5],D0,D1,P,R4
1589            **      Other R registers untouched
1590            **      Don't write to AvMem below AvMemEnd (stores BldIMG)
1591            **
1592            ** Special memory/pointer considerations (are pointers funny?):
1593            **      BldIMG tokens are stored in AvMem below BldIMG.
1594            **
1595            ** Envisioned application(s):
1596            **      1) MATH ROM uses pIMcpi poll to process complex image
1597            **         fields.  Checks that field is numeric, verifies
1598            **         that no more than 2 numeric fields are within the
1599            **         complex field.
1600            **      2) Say a LEX file implements a numeric field descrip-
1601            **         tor which encloses negative numbers in parentheses.
1602            **         The syntax might be, say, "-DDD.D", where a leading
1603            **         "-" would identify this type of descriptor.  E.g.,
1604            **             DISP USING "-3D.2D"; -36.25
1605            **             displays "( 36.25)".
1606            **         It would cause a pIMCHR poll for the "-" symbol.
1607            **         At that time, the LEX file could set S7=1, R1(S)=0.
1608            **         When the numeric field is initialized, the pIMcpi
1609            **         poll should be handled to 1) check to make sure
1610            **         it is a numeric field, 2) put appropriate execution
1611            **         tokens in the BldIMG stream to effect the right
1612            **         output, and 3) set S7=0.  Note that this new de-
1613            **         scriptor would not be allowed with complex fields,
1614            **         either imbedded inside them, or vice versa (unless
1615            **         some very creative code was written).
1616            **
1617            ** History:
1618            **
1619            **    Date     Programmer            Modification
1620            **  --------   ----------   ----------------------------------
1621            **  12/08/82   MB           Implemented, documented.
1622            **
1623            ****************************************************************
1624            ****************************************************************
```

```
1625                    EJECT
1626            ****************************************************************
1627            ****************************************************************
1628            **
1629            ** Name:(S) USING   -  Interpret IMAGE String
1630            **
1631            ** Category:   STEXEC
1632            **
1633            ** Purpose:
1634            **      Parse IMAGE stmt for formatted input/output (DISP USING,
1635            **        PRINT USING, ENTER USING, etc.)
1636            **
1637            ** Entry:
1638            **      P      = 0
1639            **      D0= program PC (points to IMAGE string or line #)
1640            **      D1 points to next item on stack.
1641            **
1642            ** Exit:
1643            **      If error (IMAGE parse or USING xqt), to MFERR.
1644            **      Otherwise, to NXTSTM, unless picked up by poll handler.
1645            **
1646            ** Calls:  EXPEXC... Need I say more?
1647            **
1648            ** Uses:   EXPEXC can use all CPU registers.
1649            **
1650            ** Stk lvls:   4  (all stack levels are lost, since the
1651            **                 IMAGE parse routines use the stack
1652            **                 for storage)
1653            **
1654            ** NOTE: All RSTK levels are lost.  Never call USING expecting
1655            **       any RSTK levels to be saved.
1656            **
1657            ** Detail:
1658            **   Register usage:
1659            **
1660            **     D0= pointer into IMAGE string.
1661            **     D1= pointer into BldIMG (expanded string where execution
1662            **           code is built)
1663            **
1664            **     D(A)= address of available memory start.
1665            **
1666            **     R0(A)  = D1 where backward search was started.
1667            **     R0(9-5)= address to start execution.
1668            **
1669            **     R1(A)  = stores D0.
1670            **     R1(9-5)= length of IMAGE string (nibs).
1671            **     R1(S)  = counter for 2 complex numeric fields.
1672            **
1673            **     R2(A)  = counter for digits in front of radix
1674            **
1675            **     R3(A)  = Program Counter (D0 at entry or re-entry).
1676            **     R3(9-5)= Address of start of IMAGE string.
1677            **
1678            **     Image tokens for building expanded IMAGE.
1679            **     1) Tokens not identifying the end of a numeric field.
```

```
1680            **      1a) Tokens not used in backwards search.
1681            **      uSTRPT =#D0     String pointer
1682            **      uSTRPT =#D0     String pointer
1683            **      uMULT  =#D1     Multiplier
1684            **      uLOOPB =#D2     Loop on byte
1685            **      uLOOPS =#D3     Loop on string (12 nibs)
1686            **      uIMXCH =#D4     Strange execution character.
1687            **
1688            **      1b) Tokens used in backwards search.
1689            **      uOPNNM =#D8     Open loop without multiplier
1690            **      uJMP{} =#D9     Jump over paren loop ptr (9 nibs)
1691            **      uJMPst =#DA     Jump over string pointer (14 nibs)
1692            **      uJMPdl =#DB     Jump over unfilled delimiter (8nibs)
1693            **      uIMbck =#DC     Poll for backward search handler
1694            **      uIMsta =#DE     IMAGE string start (|Dx| - see IMentr)
1695            **      uOPNM- =#DF     Open loop with mult, decremented
1696            **      uOPNWM =#E0     Open loop with mult  (ends in 0!)
1697            **
1698            **+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1699            **      EndNum =#E6       Any value >= this identifies the    +
1700            **                        end of a numeric field (used        +
1701            **                        in execution).                      +
1702            **+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1703            **
1704            **    2) Tokens identifying the end of a numeric field.
1705            **      2a) Tokens not used in backwards search.
1706            **      uCPLXC =#EE       Complex field closed
1707            **      uLOOPP =#EF       Loop on parentheses (variable #bytes)
1708            **      uIMend =#F0       IMAGE string end
1709            **
1710            **      2b) Tokens used in backwards search.
1711            **      uRESTP =#F1       Restart parse
1712            **      uDELIM =#F4       Delimiter
1713            **          Tokens delimiting an output/input field.
1714            **      uHKB^  =#F6       H,K,B or ^ field
1715            **      uALit  =#F7       "A" literal field
1716            **      uNUMNn =#F8       Numeric, no float chars, no sign*
1717            **      uNUMNs =#F9       Numeric, no float chars, w/sign*
1718            **      uNUMFn =#FA       Numeric, w/float chars, no sign*
1719            **      uNUMFs =#FB       Numeric, w/float chars, w/sign*
1720            **      uNUMEn =#FC       Numeric, w/Exponent, no sign*
1721            **      uNUMEs =#FD       Numeric, w/Exponent, w/sign*
1722            **
1723            **  *Note: these numeric delimiters have values that
1724            **         determine the status bit setting in USING execute.
1725            **
1726            **
1727            **   Status bits
1728            **
1729            **      These status bits must be preserved during execution!
1730            **      sMULT  =8         Multiplier pending.
1731            **      sSIGN  =9         Sign already specified.
1732            **      sFOUND =10        Output field found (at least one).
1733            **      sRDX   =11        Radix already specified.
1734            **
```

```
1735            **      These status bits can be changed during execution.
1736            **      (status bits 0,1,2 are used for numeric flags in xqt)
1737            **      sXQT    =0      Start executing.
1738            **      sC/P    =1      C/P pending.
1739            **      sCntg   =2      Counting digits.
1740            **      sInit   =3      Field already initialized.
1741            **      InhEOL  =4      Same as SB&IO !!! (always=0)
1742            **      sSTOP   =5      Stop backward search.
1743            **      sSpecl  =6      Special handling (used in xqtn)
1744            **      sCplxP  =7      Complex field pending.
1745            **
1746            **
1747            **   Bits for character masks used in parsing (CkLoop)
1748            **
1749            **      X-chr   =2^15   X: "blank"
1750            **      D-chr   =2^14   D: digit
1751            **      A-chr   =2^13   A: string char
1752            **      Pt-chr  =2^12   Decimal point
1753            **      Dblqt   =2^11   Dbl quote: literal delim
1754            **      Sglqt   =2^10   Single quote: literal delim
1755            **      S-chr   =2^9    S: sign
1756            **      M-chr   =2^8    M: sign
1757            **      Z-chr   =2^7    Z: digit
1758            **      E-chr   =2^6    E: exponent
1759            **      C-chr   =2^5    C: separator
1760            **      astrsk  =2^4    *: digit
1761            **      Z1-chr  =2^3    Z: unit's digit A
1762            **      P-chr   =2^2    P: separator
1763            **      R-chr   =2^1    R: radix
1764            **      Nf      =2^0    Nxtfld flag: return to Nxtf15.
1765            **      ed      =(X-chr)+(Dblqt)+(Sglqt)   Edit chars
1766            **      CP      =(C-chr)+(P-chr)           Separators
1767            **      SM      =(S-chr)+(M-chr)           Sign chars
1768            **      Rx      =(Pt-chr)+(R-chr)          Radix chars
1769            **      edSMRx  =(ed)+(SM)+(Rx)
1770            **      CPE     =(CP)+(E-chr)
1771            **
1772            ** Algorithm:
1773            **  1:Statement set-up:
1774            **     If IMAGE is referred to by line no.,
1775            **       establish program scope (in case keyboard xqt)
1776            **       point D1 to line# (PFINDL)
1777            **       skip over any line labels, find start of IMAGE string
1778            **       calculate IMAGE string length
1779            **       write uIMend token ("end of IMAGE string") to AvMemEnd
1780            **       move IMAGE string to AvMemEnd
1781            **       goto  2
1782            **
1783            **     If IMAGE is referred to by a string expression,
1784            **       write uIMend token to AvMemEnd,
1785            **       call EXPEXC (EXPR) to put string on stack at AvMemEnd
1786            **       reverse string so it's in "normal" direction (REVPOP)
1787            **       store D0(=PC) and D1(=start of IMAGE string) in R3.
1788            **
1789            **  2:IMAGE parse:
```

```
1790                **     Follow the parse tree laid out in individual parse
1791                **     routines.
1792                **
1793                ** History:
1794                **
1795                **    Date     Programmer            Modification
1796                **  --------   ----------   ----------------------------------
1797                **  08/10/82   MB           Started writing code
1798                **  11/10/82   MB           Finished writing code
1799                **  01/14/83   MB           Updated documentation
1800                ********************************************************************
1801                ********************************************************************
1802                *
1803 1B446          =USING
1804 1B446 161            DO=DO+ 2              Increment PC.
1805 1B449 14A            A=DAT0 B
1806 1B44C 3100           LC(2)  =tLINE#        Test for DISP USING <line#>.
1807 1B450 966            ?A#C   B              Line # for image?
1808 1B453 37             GOYES  USG007         No.
1809 1B455 136            CDOEX                 Save DO in R1.
1810 1B458 109            R1=C
1811 1B45B 8F00           GOSBVL =PRSCKB        Establish program scope.
           000
1812 1B462 77C5           GOSUB  IMD0+2         Restore DO from R1,
1813                *                              increment to ";" token.
1814 1B466 8F00           GOSBVL =PFINDL        Set D1 to line with line#.
           000
1815 1B46D 587            GONC   STMTNF         NC= Line# not found.
1816 1B470 168            DO=DO+ 9              Increment DO past line#, chain.
1817 1B473 137            CD1EX                 Switch D1 to DO, ...
1818 1B476 136            CDOEX                   save DO
1819 1B479 10B            R3=C                    in R3.
1820 1B47C 163            DO=DO+ 4              DO to IMAGE length.
1821 1B47F 5A1            GONC   USG005         (BET)
1822                *
1823 1B482 183   USG003  DO=DO- 4              Point to label length.
1824 1B485 8F00           GOSBVL =LINSKP        Skip over label.
           000
1825 1B48C 14A            A=DAT0 B              Read 1st token past label.
1826 1B48F 300            LC(1)  =tEOL          C(B)= tEOL  ( C(1) still=F).
1827 1B492 962            ?A=C   B              Null line?
1828 1B495 35             GOYES  IvUSGj         Yes. "Invalid USING".
1829 1B497 161            DO=DO+ 2              DO to length of remaining stmt.
1830                *
1831                *
1832 1B49A D2    USG005   C=0    A
1833 1B49C 306            LCHEX  6              Correction for IMAGE length.
1834 1B49F D5             B=C    A
1835 1B4A1 14E            C=DAT0 B              C= IMAGE length plus other fields.
1836 1B4A4 ED             B=C-B  A              B(A)= IMAGE length (for MOVEDO).
1837 1B4A6 161            DO=DO+ 2              DO points to IMAGE token.
1838 1B4A9 14A            A=DAT0 B              Read token.
1839 1B4AC 161            DO=DO+ 2
1840 1B4AF 3100           LC(2)  =tLBLST        Check token for starting label.
1841 1B4B3 962            ?A=C   B              Line# with label?
```

```
1842 1B4B6 CC           GOYES  USG003        Yes. Skip over label.
1843 1B4B8 B64          A=A+1  B             IMAGE token? (tIMAGE= FF)
1844 1B4BB 5C2          GONC   IvUSGj        NO! "Invalid USING".
1845 1B4BE 136          CDOEX
1846 1B4C1 C9           C=B+C  A             Add offset to DO, point to end.
1847 1B4C3 134          DO=C                 DO= end of source.
1848 1B4C6 7035 USG007  GOSUB  SetAVM        D1= AvMemEnd, D= AvMemSt.
1849 1B4CA 310F         LC(2)  uIMend        Write "Image end" token
1850 1B4CE 7695         GOSUB  BldIMG          to BldIMG string.
1851 1B4D2 96C          ?A#0   B             DISP USING <string expr> ?
1852 1B4D5 02           GOYES  USG009        Yes.
1853 1B4D7 7000         GOSUB  =MOVEDO       Move IMAGE stmt to AvMem.
1854 1B4DB 11B          C=R3                 Restore DO from R3.
1855 1B4DE 134          DO=C
1856 1B4E1 D9           C=B    A             #nibs in string.
1857 1B4E3 5C2          GONC   USG011        (BET)
1858            *
1859            ************************************
1860            *
1861 1B4E6 2E   =STMTNF P=     14            (Loads eSTMNF into C(B).)
1862            *                            Fall into IvUSGj
1863            *
1864 1B4E8      =IvUSGj                      Reads "LC(4) (eSTMNF)^(eINVUS)
1865 1B4E8 33           NIBHEX 33
1866 1B4EA 00           CON(2) =eINVUS       "Invalid USING".
1867 1B4EC 00           CON(2) =eSTMNF       "Stmt Not Found".
1868 1B4EE 8D00 =kMFERR GOVLNG =MFERR0       Sets P=0, calls MFERR.
          000
1869            *
1870            ************************************
1871            *
1872 1B4F5 8E00  USG009 GOSUBL =expr          Fetch image string expression.
          00
1873 1B4FB 8E00         GOSUBL =CKINFO       Reset output info.
          00
1874 1B501 8F00         GOSBVL =REVPOP       Reverse string; A(A)=length.
          000
1875 1B508 8E00         GOSUBL =D=AVMS       Set D=AvMemSt for BldIMG call.
          00
1876 1B50E D6           C=A    A             #nibs in string.
1877            *
1878 1B510 08    USG011 CLRST
1879 1B512 70F4         GOSUB  CSLW9j        String length to C(9-5).
1880 1B516 10A          R2=C                 Clear R2 (for IMprct).
1881 1B519 31ED         LC(2)  uIMsta
1882 1B51D 7745         GOSUB  BldIMG        Write uIMsta to start of buffer.
1883 1B521 137          CD1EX                Save addr start of IMAGE string
1884 1B524 109          R1=C                   in R1(A), length of string
1885 1B527 137          CD1EX                  in R1(9-5).
1886 1B52A 8F00         GOSBVL =R3=D10       Save D1 in R3(9-5), DO in R3(A).
          000
1887 1B531 7BD4         GOSUB  CSL9R+        Store D1=xqt addr in R0(9-5).
1888            *
```

```
1889                    EJECT
1890                ****************************************************************
1891                ****************************************************************
1892                **
1893                ** Name:    IMentr  -  Enter IMAGE string, start parsing
1894                **
1895                ** Category:   LOCAL
1896                ** Purpose:    Enter IMAGE string, start parsing; look for
1897                **             carriage control character.
1898                **
1899                ** Entry:      P=0
1900                ** Exit:       P=0. To IMprct if "#" symbol, to Nxtfld if not.
1901                ** Calls:      PRSscn
1902                **
1903                ** Uses:       C(A). A(B).
1904                **
1905                ** Stk lvls:   NA
1906                **
1907                ** Algorithm:
1908                **   1: Read in char, scan table of symbols.
1909                **      Add 2 to D0 (in R1(A)).
1910                **      If symbol= ASCII blank, goto 1.
1911                **      If symbol= "#", goto IMprct.
1912                **        Else return D0 (in R1(A)) to symbol, goto Nxtfld.
1913                **
1914                ** History:
1915                **
1916                **    Date     Programmer             Modification
1917                **  --------   ----------   ---------------------------------
1918                **  12/08/82   MB           Wrote routine, documented.
1919                **
1920                ****************************************************************
1921                ****************************************************************
1922                *  Start parsing IMAGE string
1923                *
1924 1B535 7B45  IMentr GOSUB  PRSsc+          Scan table of tokens.
1925                *
1926 1B539 32          CON(2) \#\
1927 1B53B 110         REL(3) IMprct          Suppress ENDLINE.
1928                *
1929 1B53E 02          CON(2) \ \
1930 1B540 5FF         REL(3) IMentr          Space ignored.
1931                *
1932 1B543 00          CON(2) 0
1933 1B545 78D4        GOSUB  IMD0-2          D0 back to current char.
1934 1B549 537         GONC   Nxtfld          (BET)
1935                *
1936                ************************************
1937                *
1938 1B54C 853   IMprct ST=1   sInit           To pass through test at CkDlim.
1939 1B54F 7611        GOSUB  CkLoop          Check for spaces.
1940 1B553 0000        CON(4) 0
1941                *
```

```
1942                    EJECT
1943            ****************************************************************
1944            ****************************************************************
1945            **
1946            ** Name:    CkDlBS  -  Byte scan for delimiter check
1947            **
1948            ** Category:   LOCAL
1949            ** Purpose: Subroutine provides byte scan for delimiter check.
1950            ** Entry:      P=0
1951            ** Exit:       To appropriate processor for delimiter, or
1952            **               to poll if unrecognized symbol.
1953            **
1954            ** Calls:      PRSscn
1955            **
1956            ** Uses..... Could use anything in delimiter routines.
1957            **
1958            ** Stk lvls:   NA
1959            **
1960            ** Algorithm:
1961            **    Read char from IMAGE string, scan byte table for jump.
1962            **    If character not in table, poll: pIMCHR.
1963            **      If poll not handled, "Invalid IMAGE" error.
1964            **
1965            ** History:
1966            **
1967            **    Date     Programmer             Modification
1968            **  --------   ----------   ----------------------------------
1969            **  12/08/82   MB           Wrote routine, documented.
1970            **
1971            ****************************************************************
1972            ****************************************************************
1973            *
1974 1B557 7D25  CkDlBS GOSUB  PRSscn         Byte scan for CkDlim.
1975            *
1976 1B55B C2           CON(2) \,\
1977 1B55D 304          REL(3) IM","         Comma delimiter.
1978            *
1979 1B560 F2           CON(2) \/\
1980 1B562 2E3          REL(3) IMmult        Send ENDLINE.
1981            *
1982 1B565 04           CON(2) \@\
1983 1B567 DD3          REL(3) IMmult        Send FORM FEED.
1984            *
1985 1B56A 92           CON(2) \)\           (Right parenthesis)
1986 1B56C E82          REL(3) IM"}"         Close parenthesis loop.
1987            *
1988 1B56F 82           CON(2) \(\
1989 1B571 D74          REL(3) IM"("         Open parenthesis loop.
1990            *
1991 1B574 0F           CON(2) uIMend
1992 1B576 1F3          REL(3) IMGend        IMAGE string end.
1993            *
1994 1B579 00           CON(2) 0                 (Invalid Char)
1995 1B57B 7414  IMCHRp GOSUB  FPOLLx        Invalid IMAGE char poll.
1996 1B57F 00           CON(2) =pIMCHR
```

```
1997 1B581 7104          GOSUB  Polrtn         Check for poll handled.
1998             *         +-----------------------------------------+
1999             *         |  If poll handled, handler must return:  |
2000             *         |     D1 restored (from R0(A))            |
2001             *         |     D(A)= AvMemSt                       |
2002             *         |     P as indicated below, for jump.     |
2003             *         +-----------------------------------------+
2004 1B585 0D            P=P-1
2005 1B587 453           GOC    Nxtfld         P=0: "Rtn to Nxtfld".
2006 1B58A 890           ?P=    0
2007 1B58D 60            GOYES  CkDlm+         P=1: "Check for delimiter".
2008             *                 !!!!! To go to CkDlm+, poll handler
2009             *                 !!!!! must set sInit=1      !!!!!!!
2010 1B58F 6000          GOTO   =IMGxq1        P=2: "Execute IMAGE field."
2011             *
```

```
2012                    EJECT
2013                ****************************************************************
2014                ****************************************************************
2015                **
2016                ** Name:    CkDlim  -  Check IMAGE string for delimiter
2017                **
2018                ** Category:   LOCAL
2019                ** Purpose:    Check IMAGE string for delimiter.
2020                ** Entry:      P=0
2021                ** Exit:       To Nxtfld
2022                **
2023                ** Calls:      CkDlm+ calls CkEdi
2024                **             CkDlim call WR#DG3  and  CkDlBS
2025                **
2026                ** Stk lvls:   NA
2027                **
2028                ** Algorithm:
2029                **   CkDlm+: Check for edit chars before the delimiter (CkEdit)
2030                **             If there are any, CkEdit puts them in BldIMG.
2031                **   CkDlim: If C/P pending, error (< or P not allowed before
2032                **             a delimiter).
2033                **           If adjacent delimiters, back up over previous one
2034                **             (simply saves RAM).
2035                **           If counting digits (i.e., just finished a numeric
2036                **             field), write out digit count to BldIMG (WR#DG3).
2037                **           Byte scan of next IMAGE character (CkDlBS),
2038                **             process next delimiter, return to Nxtfld.
2039                **
2040                ** History:
2041                **
2042                **    Date     Programmer              Modification
2043                **  --------   ----------   ----------------------------------
2044                **  12/08/82   MB           Wrote routine, documented.
2045                **
2046                ****************************************************************
2047                ****************************************************************
2048                *
2049 1B593 76D0  CkDlm+ GOSUB  CkLpNC          Ready for delim.
2050 1B597 00C8        CON(4) ed              Check for edit chars only.
2051                *
2052 1B59B 863   CkDlim ?ST=0  sInit           Back from CkLoop; error?
2053 1B59E DD          GOYES  IMCHRp          Yes. Poll for unrecognized char.
2054 1B5A0 32BD        LC(3)  #F~(uJMPdl)     (Extra F for WR#DIG below.)
           F
2055 1B5A5 14B         A=DAT1 B
2056 1B5A8 966         ?A#C   B               Double delimiter?
2057 1B5AB 50          GOYES  CkDl03          No.
2058 1B5AD 179         D1=D1+ 10              Yes, back up to previous one.
2059 1B5B0 862   CkDl03 ?ST=0  sCntg          Counting digits?
2060 1B5B3 60          GOYES  CkDl07          No.
2061 1B5B5 7AE3        GOSUB  WR#DG3          Back to delim, fill in #digits.
2062                *
2063 1B5B9 7A9F  CkDl07 GOSUB  CkDlBS          Byte scan.
2064                *                         Fall through to Nxtfld.
```

```
2065                    EJECT
2066              ********************************************************************
2067              ********************************************************************
2068              **
2069              ** Name:     Nxtfld  -  Open the next IMAGE field
2070              ** Name:     Nxtfl3  -  Open the next IMAGE field
2071              **
2072              ** Category:   LOCAL
2073              ** Purpose:    Process next IMAGE field.
2074              **
2075              ** Stk lvls:   NA
2076              **
2077              ** Algorithm:
2078              **   Nxtfld: Prepare for next field by writing uDELIM
2079              **      ("unfilled delimiter") to BldIMG.
2080              **   Nxtfl3: Clear all IMAGE status bits. (CLOST+)
2081              **      Check for edit chars, put them in BldIMG (CkEdit)
2082              **      Byte scan of character table, jump to processor.
2083              **      If char not in table, it might be a delimiter -- goto
2084              **        CkDlim to check for delimiter.
2085              **
2086              ** History:
2087              **
2088              **    Date     Programmer              Modification
2089              **  --------   ----------   ---------------------------------
2090              **  12/08/82   MB           Wrote routine, documented.
2091              **
2092              ********************************************************************
2093              ********************************************************************
2094              *
2095 1B5BD 20     Nxtfld P=     0             Next IMAGE field after delim.
2096 1B5BF 849          ST=0   sSIGN         "No sign specified."
2097 1B5C2 84B          ST=0   sRDX          "No radix found."
2098 1B5C5 AF2          C=0    W
2099 1B5C8 31BD         LC(2)  uJMPdl        "Jump over unfilled delimiter".
2100 1B5CC 28           P=     8
2101 1B5CE 314F         LC(2)  uDELIM        Write Delim tokens.
2102 1B5D2 7294         GOSUB  BldIMG
2103              *
2104 1B5D6       =Nxtfl3
2105 1B5D6 70F3         GOSUB  CLOST+        Clear 0 nib of status bits, plus
2106              *                             sXQT (these bits may have been
2107              *                             set in execution routines).
2108 1B5DA 102          R2=A                 Clear digit count (A=0 in CLOST+).
2109 1B5DD 7C80         GOSUB  CkLpNC        Check for A,D,Z,*,S,M,.,R,edit
2110 1B5E1 39FF         CON(4) (Nf)+(edSMRx)+(A-chr)+(D-chr)+(Z-chr)+(astrsk)
2111              *
```

```
2112                    EJECT
2113            ********************************************************************
2114            ********************************************************************
2115            **
2116            ** Name:    IMi"A"  -  Process "A" symbol in IMAGE field.
2117            **
2118            ** Category:   LOCAL
2119            ** Purpose:    Process "A" symbol in IMAGE field.
2120            ** Entry:      P=0, D0 (in R1(A)) points to "A", D1 to BldIMG.
2121            ** Exit:       When a char is found which is not an A or not
2122            **               not an edit symbol, exit to CkDlim.
2123            ** Calls:      IMinit, CkEdit.
2124            **
2125            ** Algorithm:
2126            **    1:  Initialize field (change unfilled delimiter) to
2127            **          a literal field (IMinit).
2128            **    2:  Check for pending multiplier; if there is, fill out
2129            **          multiplier fields. In any case, write "A" symbol
2130            **          to BldIMG.
2131            **        Byte scan for another "A"; if found, goto 2.
2132            **          Else (no "A"), go check for delimiter (CkDlim)
2133            **
2134            ** History:
2135            **
2136            **    Date     Programmer              Modification
2137            **  --------   ----------   ---------------------------------
2138            **  12/08/82   MB           Wrote routine, documented.
2139            **
2140            ********************************************************************
2141            ********************************************************************
2142            *
2143 1B5E5       IMi"A"                        Initial "A" specifier.
2144 1B5E5       IM"A"                         "A" specifier: string char.
2145 1B5E5 317F        LC(2)  uALit
2146 1B5E9 7870        GOSUB  IMLoop          Check for A symbol,edit.
2147 1B5ED 00CA        CON(4) (ed)+(A-chr)
2148            *
2149            *
```

```
2150                    EJECT
2151                ********************************************************************
2152                ********************************************************************
2153                **
2154                ** Name:    IM1"D"  -  Process "D" character in IMAGE string
2155                **
2156                ** Category:   LOCAL
2157                ** Purpose:    Process D character in IMAGE string.
2158                ** Exit:       To IMnend (IMAGE numeric field end).
2159                **
2160                ** Calls:      IMinnu, IMmlt+, CkCPSM
2161                **
2162                ** Algorithm:
2163                **   1:   Initialize field (change unfilled delimiter) to
2164                **          uNUMFn type (numeric fld, floating, no sign).
2165                **   2:   Increment digit count, check for multiplier.  If
2166                **          there is a multiplier, set up mult fields in
2167                **          BldIMG.  In any case, write D symbol to BldIMG.
2168                **        Check for edit chars and C,P,S or M symbols.  If
2169                **          any are found (and are valid), put them in BldIMG.
2170                **        Byte scan for:
2171                **           D -- goto 2.
2172                **           Z -- (unit's digit Z) goto IM1"Z"
2173                **           other -- exit through IMnend.
2174                **
2175                ** History:
2176                **
2177                **    Date     Programmer             Modification
2178                **   --------   ----------   ----------------------------------
2179                **   12/08/82   MB           Wrote routine, documented.
2180                **
2181                ********************************************************************
2182                ********************************************************************
2183                *
2184 1B5F1          IM1"D"                      Initial "D" specifier.
2185 1B5F1          IM"D"
2186 1B5F1 87B             ?ST=1  sRDX          D after radix?
2187 1B5F4 E0              GOYES  IMDrdx        Yes.
2188 1B5F6 31AF            LC(2)  uNUMFn
2189 1B5FA 7760            GOSUB  IMLoop        Check for E,.,R,D,Z1,C,P,S,M,edit.
2190 1B5FE E6FD            CON(4) (edSMRx)+(CPE)+(D-chr)+(Z1-chr)
2191                *
2192                ************************************************************
2193                *
2194 1B602 31A5  IMDrdx LCASC  \Z\
2195 1B606 DA              A=C    A             D after radix works like Z!
2196 1B608 7550            GOSUB  IMLpnu        Check for D,C,P,S,M,E,edit
2197 1B60C 46FC            CON(4) (ed)+(SM)+(CPE)+(D-chr)
2198                *
2199                *
```

```
2200                    EJECT
2201                ****************************************************************
2202                ****************************************************************
2203                **
2204                ** Name:    IMi"*"  -  Process "*" symbol in IMAGE string
2205                **
2206                ** Category:   LOCAL
2207                ** Purpose:    Process "*" symbol in IMAGE string.
2208                ** Exit:       To IMnend (numeric IMAGE field end)
2209                ** Calls:      IMinn+, IMmlt+
2210                **
2211                ** Algorithm:
2212                **    1:   Initialize field (unfilled delimiter) with uNUMNn
2213                **           (numeric IMAGE field, non-float, no sign).
2214                **    2:   Increment digit count, check for multiplier.  If
2215                **           there is a multiplier, fill in mult fields in
2216                **           BldIMG.  In any case, write "*" symbol to BldIMG.
2217                **         Check for edit chars and C,P,S or M symbols.  If
2218                **           any are found (and are valid), put them in BldIMG.
2219                **         Byte scan for:
2220                **           * -- goto 2
2221                **           Z -- (unit's digit Z) goto IM1"Z"
2222                **           other -- goto IMnend
2223                **
2224                ** History:
2225                **
2226                **    Date     Programmer             Modification
2227                **  --------   ----------   ----------------------------------
2228                **  12/08/82   MB           Wrote routine, documented.
2229                **
2230                ****************************************************************
2231                ****************************************************************
2232                *
2233 1B610           IMi"*"                      Initial "*" specifier.
2234 1B610           IM"*"                       "*" specifier.
2235 1B610 7D40            GOSUB  IMLpnu         Check for E,.,R,*,Z1,C,P,S,M,edit
2236 1B614 E7F9            CON(4) (edSMRx)+(CPE)+(astrsk)+(Z1-chr)
2237                *
```

```
2238                    EJECT
2239                ****************************************************************
2240                ****************************************************************
2241                **
2242                ** Name:    IM"Z"   -  Process "Z" symbol in IMAGE string.
2243                **
2244                ** Category:   LOCAL
2245                ** Purpose:    Process "Z" symbol in IMAGE string.
2246                ** Exit:       To IMnend (numeric IMAGE field end)
2247                ** Calls:      IMinn+, IMmlt+
2248                **
2249                ** Algorithm:
2250                **    1:   Initialize field (unfilled delimiter) with uNUMNn
2251                **           (numeric IMAGE field, non-float, no sign).
2252                **    2:   Increment digit count, check for multiplier.  If
2253                **           there is a multiplier, fill in mult fields in
2254                **           BldIMG.  In any case, write "*" symbol to BldIMG.
2255                **         Check for edit chars and C,P,S or M symbols.  If
2256                **           any are found (and are valid), put them in BldIMG.
2257                **         Byte scan for:
2258                **           Z -- goto 2
2259                **           other -- goto IMnend
2260                **
2261                ** History:
2262                **
2263                **    Date     Programmer             Modification
2264                **  --------   ----------   ----------------------------------
2265                **  12/08/82   MB           Wrote routine, documented.
2266                **
2267                ****************************************************************
2268                ****************************************************************
2269                *
2270 1B618           IMi"Z"                        Initial "Z" specifier.
2271 1B618           IM"Z"
2272 1B618 7540            GOSUB  IMLpnu          Check for E,.,R,Z,C,P,S,M
2273 1B61C 6EF9            CON(4) (edSMRx)+(CPE)+(Z-chr)
2274                *
```

```
2275                    EJECT
2276                ********************************************************************
2277                ********************************************************************
2278                **
2279                ** Name:    IM1"Z"  -  Process unit's digit Z in IMAGE string
2280                **
2281                ** Category:   LOCAL
2282                ** Purpose:    Process unit's digit Z
2283                ** Exit:       To IMnen3.
2284                ** Calls:      IM"," , IMmlt+
2285                **
2286                ** Algorithm:
2287                **    If multiplier pending, error. (IM",")
2288                **    Increment digit count, write Z symbol to BldIMG.
2289                **    Exit through IMnen3 to catch E or radix symbols.
2290                **
2291                ** History:
2292                **
2293                **    Date     Programmer              Modification
2294                **  --------   ----------   ----------------------------------
2295                **  12/08/82   MB           Wrote routine, documented.
2296                **
2297                ********************************************************************
2298                ********************************************************************
2299                *
2300 1B620           IM1"Z"                          One's digit Z.
2301 1B620 7C33           GOSUB  IM","               Check for sMULT=0.
2302 1B624 7140           GOSUB  CkLoop              Check for E,.,R,S,M,edit
2303 1B628 24F9           CON(4) (edSMRx)+(E-chr)
2304                *
2305                *       EJECT
2306                ********************************************************************
2307                ********************************************************************
2308                **
2309                ** Name:    IMrdx   -  Process radix in IMAGE string
2310                **
2311                ** Category:   LOCAL
2312                ** Purpose:    Process radix imbedded in numeric field.
2313                ** Exit:       To CkDlim
2314                ** Calls:      WR#DIG, IMmlt+, CkCPSM
2315                **
2316                ** Algorithm:
2317                **    1:  Write symbol to BldIMG.  Fill in mantissa digit
2318                **          count. (WR#DIG)
2319                **        Set sC/P=1 to disallow C or P adjacent to radix.
2320                **        Goto 3.
2321                **    2:  Set sRDX (radix digits found)
2322                **        Change "D" symbol to "Z" (don't suppress zeros!)
2323                **        Increment digit count. If multiplier pending,
2324                **          fill out multiplier fields in BldIMG.  In any
2325                **          case, write "Z" symbol to BldIMG.
2326                **    3:  Check for C,P,S or M symbols. If there (and if
2327                **          valid), write to BldIMG.
2328                **        Byte scan for:
2329                **          D -- goto 2 (another digit following radix)
```

```
2330            **          E -- goto  IM"E"+ (exponent specified)
2331            **          other -- if radix digits found, clear sC/P
2332            **             (otherwise, the next delimiter will think
2333            **             that a C or P is pending). Goto CkDlim.
2334            **
2335            ** History:
2336            **
2337            **    Date      Programmer              Modification
2338            **  --------    ----------   ----------------------------------
2339            **  12/08/82    MB           Wrote routine, documented.
2340            **
2341            ********************************************************************
2342            ********************************************************************
2343            *
2344 1B62C       IMrdx                          Radix specifier.
2345 1B62C D2          C=0     A
2346 1B62E CE          C=C-1   A                 C(XS)#0 for WR#DIG flag.
2347 1B630 7963        GOSUB   WR#DIG            Write out before-radix count.
2348 1B634 85B         ST=1    sRDX              "Radix found".
2349 1B637 7230        GOSUB   CkLpNC            Check for S,M,D,E,edit
2350 1B63B 04FC        CON(4)  (ed)+(SM)+(D-chr)+(E-chr)
2351            *
```

```
2352                    EJECT
2353                ****************************************************************
2354                ****************************************************************
2355                **
2356                ** Name:    IMisgn  -  Process initial sign symbol in IMAGE
2357                **
2358                ** Category:   LOCAL
2359                ** Purpose:    Process initial sign in numeric IMAGE field.
2360                ** Exit:       To appropriate digit symbol processor, or error.
2361                ** Calls:      NoMult, CkEdit
2362                **
2363                ** Algorithm:
2364                **    Set sSIGN=1 ("sign already specified").
2365                **    Write symbol to BldIMG. Error if multiplier pending.
2366                **    Check for edit symbols; if there, write out to BldIMG.
2367                **    Byte scan for:
2368                **      D -- goto IMi"D" (initial D)
2369                **      Z -- goto IMi"Z" (initial Z)
2370                **      * -- goto IMi"*" (initial *)
2371                **      . -- goto IMirdx (initial .)
2372                **      R -- goto IMirdx (initial R)
2373                **      other -- pIMCHR poll
2374                **
2375                ** History:
2376                **
2377                **    Date     Programmer              Modification
2378                **   --------   ----------   ----------------------------------
2379                **   12/08/82   MB           Wrote routine, documented.
2380                **
2381                ****************************************************************
2382                ****************************************************************
2383                *
2384 1B63F           IMisgn                          Initial sign, "S" or "M".
2385 1B63F 7A20            GOSUB  CkLpNC              Check for D,Z,*,.,R,edit
2386 1B643 29CD            CON(4) (ed)+(Rx)+(Z-chr)+(astrsk)+(D-chr)
2387                *
```

```
2388                    EJECT
2389            ****************************************************************
2390            ****************************************************************
2391            **
2392            ** Name:    IMsign  -  Process sign symbol in IMAGE string
2393            **
2394            ** Category:   LOCAL
2395            ** Purpose:    Process sign symbol in numeric IMAGE field.
2396            ** Entry:      From CkCPSM
2397            ** Exit:       To CkMult
2398            **
2399            ** Calls:      NoMult, IMin01
2400            **
2401            ** Algorithm:
2402            **    If sign symbol already found (sSIGN=1), then error.
2403            **    Otherwise, set sSIGN=1 ("Sign symbol found").
2404            **    If multiplier pending, error. Otherwise, write sign
2405            **      symbol to BldIMG (NoMult).
2406            **    Re-initialize field to indicate "numeric field with
2407            **      sign specified".
2408            **    Exit to CkMult.
2409            **
2410            ** History:
2411            **
2412            **    Date     Programmer              Modification
2413            **  --------   ----------   ----------------------------------
2414            **  12/08/82   MB           Wrote routine, documented.
2415            **
2416            ****************************************************************
2417            ****************************************************************
2418            *
2419 1B647       IMsign
2420 1B647 879          ?ST=1  sSIGN          Yes. Sign already specified?
2421 1B64A 37           GOYES  IMerrc         Yes.
2422            *
2423 1B64C 859          ST=1   sSIGN          No. Set flag.
2424 1B64F 7903         GOSUB  NoMult         Write to BldIMG.
2425 1B653 863          ?ST=0  sInit          Field initialized?
2426 1B656 9E           GOYES  IMisgn         No. Initial sign specifier.
2427 1B658 D2           C=0    A              For IMin01.
2428 1B65A 7842         GOSUB  IMin01         Re-write delim "with sign."
2429 1B65E 5E0          GONC   CkMult         (BET)
2430            *
```

```
2431                    EJECT
2432            ********************************************************************
2433            ********************************************************************
2434            **
2435            ** Name:(S) CkLoop  -  IMAGE parse loop to check for edit chars
2436            ** Name:(S) CkLpNC  -  IMAGE parse loop, no symbol count
2437            **
2438            ** Category:   EXCUTL
2439            **
2440            ** Purpose:
2441            **      This is the main parsing routine for IMAGE parsing.
2442            **      It first accepts spaces and multipliers in the image
2443            **      string, then parses the next image character for
2444            **      correct syntax.
2445            **
2446            ** Entry:
2447            **      P      = 0
2448            **      R1(A) points to last IMAGE symbol which was parsed
2449            **      D1 points to current position in BldIMG stream
2450            **      D(A)=AvMemEnd
2451            **      Return address contains a four-nibble mask used
2452            **        to parse the next character (see USING header)
2453            **
2454            ** Exit:
2455            **      CkLoop does not return!  It jumps to
2456            **          1) IMerr (if multiplier=0 is found)
2457            **          2) CkDlim (if no match found in parse table)
2458            **          3) To appropriate parse routine if match found
2459            **              (these routines are fixed in the parse table;
2460            **               they cannot be added to)
2461            **      CkLoop leaves the RSTK in a mess...  The parse mask
2462            **      address is left in the RSTK (no problem, since USING
2463            **      can never be called as a subroutine).
2464            **
2465            ** Calls:      IMmlt+, PRSsc+, DRANGE, TBLJMP
2466            **
2467            ** Uses.......
2468            **  Exclusive: A,B,C,D,D0,D1,P
2469            **  Inclusive: Can use anything when exits to parse handlers
2470            **
2471            ** Stk lvls:   3 (before exit to parse handler)
2472            **
2473            ** NOTE:
2474            **      This parser is used only for the following IMAGE sym-
2475            **      bols, with the corresponding parse handler routines:
2476            **          X                   IM"X"
2477            **          D                   IM"D"
2478            **          A                   IM"A"
2479            **          . or R              IMrdx
2480            **          " or '              IMstr
2481            **          S or M              IMsign
2482            **          Z                   IM"Z"
2483            **          E                   IM"E"
2484            **          C or P              IMsep
2485            **          *                   IM"*"
```

```
2486            **          unit's digit Z      IM1"Z"
2487            **          H,K,B or ^          IMHKB^
2488            **      Only those symbols included in the parse mask (found
2489            **      at the RSTK address) will be accepted.  Any other
2490            **      character will cause a jump to CkDlim; if the char-
2491            **      acter is not a delimiter, CkDlim will issue a pIMCHR
2492            **      poll.
2493            **
2494            ** Algorithm:
2495            **   CkLoop: Increment digit symbol count (in R2(A))
2496            **   CkLpNC:
2497            **     1)    Fetch next IMAGE character, convert to uppercase
2498            **           Check for digit (DRANGE); if not digit, goto 3).
2499            **           Else (digit) if digit already found (sMULT=1)
2500            **              goto 2).
2501            **           Else (digit not found yet) set sMULT=1,
2502            **              write out multiplier fields to BldIMG.
2503            **     2)    Check for digit overflow (more than 4 digits),
2504            **              error if overflow.
2505            **           Write out current multiplier.
2506            **           Goto 1).
2507            **     3)    Check char for ASCII space. If so, goto 1).
2508            **           If multiplier pending, test for:
2509            **              if mult= 0, then error.
2510            **              if mult= 1, then ignore (back up over
2511            **                   multiplier fields)
2512            **           Fetch parse mask from RSTK address.
2513            **     4)    Read next character from fixed parse table.
2514            **           If end of table, jump to CkDlim.
2515            **           Check mask bit for valid char; if not,
2516            **             go to 4).
2517            **           Else (valid char), compare with IMAGE symbol:
2518            **             if no match, go to 4)
2519            **             else (match), jump to parse handler for
2520            **               that symbol
2521            **
2522            ** History:
2523            **
2524            **    Date     Programmer             Modification
2525            ** --------   ----------   ---------------------------------
2526            **  12/08/82   MB          Wrote routine, documented.
2527            **
2528            ************************************************************
2529            ************************************************************
2530            *
2531 1B661 318F  IMLpnu LC(2)  uNUMNn         "Non-floating numeric field".
2532 1B665 7622  IMLoop GOSUB  IMinit         Initialize field.
2533            *
2534 1B669      =CkLoop                      Check loop: count symbol, ck edit,
2535            *                            parse scan for designated symbols.
2536 1B669 74B2         GOSUB  IMmlt+        Count symbol.
2537            *
2538 1B66D      =CkLpNC                      Check loop, no symbol count.
2539 1B66D 7314  CkMult GOSUB  PRSsc+        Next char from DATO, upper case.
2540 1B671 00           CON(2) 0             (Does a FINDA)
```

```
2541 1B673 7000          GOSUB  =DRANGE       Digit?
2542 1B677 423           GOC    CkM115        No.
2543 1B67A 878           ?ST=1  sMULT         Yes. First digit?
2544 1B67D 71            GOYES  CkM103        No.
2545 1B67F 858           ST=1   sMULT         Set "multiplier pending" flag.
2546               *
2547 1B682 39            NIBHEX 39            Reads:
2548 1B684 0000          CON(4) 0000           LC(10) xx00xx0000
2549 1B688 0E            CON(2) uOPNWM                 /   /
2550 1B68A 00            CON(2) 00                uMULT  /
2551 1B68C 1D            CON(2) uMULT                  uOPNWM
2552               *
2553 1B68E 29            P=     9
2554 1B690 76D3          GOSUB  BldIM+        Write mult token to BldIMG.
2555               *
2556               *
2557 1B694 AF2   CkM103  C=0    W             Clear upper C(M).
2558 1B697 147           C=DAT1 A             Read existing multiplier
2559               *                             (mult=0 at first)
2560               *                             C(4)=0 since uOPNWM=E0.
2561 1B69A 95E           ?C#0   M             Overflow? (max= 9999) (C(3)#0?)
2562 1B69D 02            GOYES  IMerrc        Yes.
2563               *
2564 1B69F F2            CSL    A             x10. (C(4)=0 since passed test).
2565 1B6A1 A86           C=A    P             Copy new digit.
2566 1B6A4 145           DAT1=C A             Write out current multiplier.
2567 1B6A7 55C           GONC   CkMult        (BET) Check next char for digit.
2568               *
2569 1B6AA 3102  CkM115  LCASC  \ \
2570 1B6AE 962           ?A=C   B             Imbedded space in multiplier?
2571 1B6B1 CB            GOYES  CkMult        Yes. Check for more digits.
2572 1B6B3 868           ?ST=0  sMULT         Multiplier pending?
2573 1B6B6 51            GOYES  CkM117        No.
2574 1B6B8 147           C=DAT1 A             Yes. Copy mult, test for 0 or 1.
2575               *                             (C(4)=0 since uOPNWM=E0.)
2576 1B6BB CE            C=C-1  A             Zero?
2577 1B6BD 4A4   IMerrc  GOC    IMerrd        Yes.
2578               *
2579 1B6C0 8AE           ?C#0   A             No. 1?
2580 1B6C3 80            GOYES  CkM117        No. Valid multiplier.
2581 1B6C5 848           ST=0   sMULT         Yes. Cancel multiplier.
2582 1B6C8 179           D1=D1+ 10            Restore D1 before multiplier.
2583 1B6CB 07    CkM117  C=RSTK               Table address-3.
2584 1B6CD 06            RSTK=C               Restore return address.
2585 1B6CF 134           D0=C
2586 1B6D2 146           C=DAT0 A             Read mask to C(3-0).
2587 1B6D5 F2            CSL    A             Mask to C(4-1).
2588 1B6D7 D5            B=C    A             Copy table, flags to B.
2589 1B6D9 1B00          D0=(5) =IMtbl        Point to Image table.
           000
2590               *
2591 1B6E0 14E   CkM121  C=DAT0 B             Read Image char.
2592 1B6E3 161           D0=D0+ 2             To next table char.
2593 1B6E6 C5            B=B+B  A             Char allowed in mask?
2594 1B6E8 561           GONC   CkM125        No.
```

```
2595 1B6EB 962          ?A=C   B            Yes. Match?
2596 1B6EE E1           GOYES  CkM127        Yes! Execute char.
2597 1B6F0 88F          ?P#    15           Nxtflf bit set?
2598 1B6F3 C0           GOYES  CkM125        No. Check next char in table.
2599 1B6F5 CD           B=B-1  A            Yes. Set all bits in B(A).
2600 1B6F7 96E          ?C#0   B            End of extended table?
2601 1B6FA 6E           GOYES  CkM121        No.
2602 1B6FC 853          ST=1   sInit        Yes. ST=1 to pass test at CkDlim.
2603 1B6FF 0C    CkM125 P=P+1                End of table?
2604 1B701 5ED          GONC   CkM121        No.
2605 1B704 669E         GOTO   CkDlim        Yes, no match. Check for delim.
2606            *
2607            *
2608 1B708 6082  IMerrd GOTO   IMerr
2609            *
2610            *
2611 1B70C 8F00  CkM127 GOSBVL =TBLJMP       Jump on table with P=indicator.
         000
2612            *                          P=
2613 1B713 5C0          REL(3) IM"X"          0: X    (X must come first!)
2614 1B716 BDE          REL(3) IM"D"          1: D
2615 1B719 CCE          REL(3) IM"A"          2: A
2616 1B71C 01F          REL(3) IMrdx          3: .
2617 1B71F 420          REL(3) IMstr          4: "
2618 1B722 120          REL(3) IMstr          5: '
2619 1B725 22F          REL(3) IMsign         6: S
2620 1B728 F1F          REL(3) IMsign         7: M
2621 1B72B DEE          REL(3) IM"Z"          8: Z
2622 1B72E A80          REL(3) IM"E"          9: E
2623 1B731 EA0          REL(3) IMsep         10: C
2624 1B734 CDE          REL(3) IM"*"         11: *
2625 1B737 9EE          REL(3) IM1"Z"        12: unit's digit Z
2626 1B73A 5A0          REL(3) IMsep         13: P
2627 1B73D FEE          REL(3) IMrdx         14: R
2628 1B740 BA0          REL(3) IMHKB^        15: H,K,B or ^.
2629            *
2630            *
```

```
2631                    EJECT
2632            ****************************************************************
2633            ****************************************************************
2634            **
2635            ** Name:    IMstr   -  Process imbedded literal in IMAGE string
2636            **
2637            ** Category:   LOCAL
2638            ** Purpose:    Process imbedded literal in IMAGE string.
2639            ** Entry:      From CkEdit
2640            ** Exit:       Back to CkEdit
2641            **
2642            ** Calls:      CSRWP9, IMoffs, BldIM+
2643            **
2644            ** Uses....... A,B,C  ...
2645            **
2646            ** Stk lvls:   NA
2647            **
2648            ** Algorithm:
2649            **    Copy D0 (=addr of " or ') to B(A).
2650            **    Length of IMAGE string to C(A).
2651            **    Set D0= addr of first char in imbedded literal.
2652            **    Write uSTRPT token to BldIMG.
2653            **    Store offset to imbedded literal in BldIMG.
2654            **    Address of IMAGE string start to C(A).
2655            **    Set B=B+C= max possible length of literal.
2656            **    Scan literal for matching delimiter, counting
2657            **      characters in C(M), decrementing B(A) so that
2658            **      the IMAGE string is not exceeded.
2659            **    Literal count (length) to C(6-2); write length
2660            **      and uJMPst token to BldIMG.
2661            **    Now that D0 points past imbedded literal, set
2662            **      R1(A) to new D0.
2663            **    If multiplier pending, copy multiplier reserve
2664            **      into multiplier counter, write uLOOPS token
2665            **      (loop back to literal multiplier) to BldIMG.
2666            **    Exit to CkMult
2667            **
2668            ** History:
2669            **
2670            **    Date     Programmer             Modification
2671            **  --------   ----------   ----------------------------------
2672            **  12/08/82   MB           Wrote routine, documented.
2673            **
2674            ****************************************************************
2675            ****************************************************************
2676            *
2677 1B743       IMstr                          Scan literal string.
2678            *             Max length possible for literal
2679            *                = (length of IMAGE string) - ( (D0 - start ) )
2680            *                = (length of IMAGE string) - D0 + start
2681            *
2682 1B743 AF1          B=0    W               (For BSRB below.)
2683 1B746 119          C=R1                   C(A)= D0.
2684 1B749 D5           B=C    A               D0 to B.
2685 1B74B 8E00         GOSUBL =CSRW9j         C(A)= Length of IMAGE string.
```

```
            00
2686 1B751 ED           B=C-B  A            B(A)= length - DO.
2687 1B753 76D2         GOSUB  IMDO+2       Fetch DO, increment to next char.
2688 1B757 25           P=     5
2689 1B759 310D         LC(2)  uSTRPT       String pointer token.
2690 1B75D 77F2         GOSUB  IMoffs       Offset to string to BldIMG.
2691 1B761 11B          C=R3
2692 1B764 8E00         GOSUBL =CSRW9j      C(A)= addr start IMAGE string.
            00
2693 1B76A C1           B=B+C  A            B= max length of literal string.
2694 1B76C 81D          BSRB                B= max length in bytes.
2695 1B76F D2           C=0    A            C(M) for counting literal length.
2696              *
2697 1B771 CD    IMstr3 B=B-1  A          Count #chars available for literal.
2698 1B773 449   IMerrb GOC    IMerrd       Quote not found.
2699              *
2700 1B776 14E          C=DATO B            Read next char in literal.
2701 1B779 962          ?A=C   B            Delimiter?
2702 1B77C B0           GOYES  IMstr7       Yes. End of literal.
2703 1B77E 161          DO=DO+ 2            No. To next char.
2704 1B781 B56          C=C+1  M            Count chars.
2705 1B784 5CE          GONC   IMstr3       (BET) Test next char.
2706              *
2707              *                  Note: no need to check length of literal
2708              *                    since B(A)<FFFF.
2709 1B787 BF6   IMstr7 CSR    W            Length of literal to C(6-2).
2710 1B78A 31AD         LC(2)  uJMPst       "Jump over string pointer" token.
2711 1B78E 26           P=     6
2712 1B790 76D2         GOSUB  BldIM+       Write length, token to BldIMG.
2713 1B794 119          C=R1                Preserve R1(9-5).
2714 1B797 136          CDOEX               New DO to R1(A).
2715 1B79A 109          R1=C
2716 1B79D 868          ?ST=0  sMULT        Multiplier pending?
2717 1B7A0 41           GOYES  CkMltj       No.
2718 1B7A2 17D          D1=D1+ 14           Back to multiplier storage.
2719 1B7A5 7F22         GOSUB  COPYmu       Copy multiplier into loop counter.
2720 1B7A9 1CD          D1=D1- 14           Back to current D1.
2721 1B7AC 313D         LC(2)  uLOOPS       "Loop on string" token.
2722 1B7B0 74B2         GOSUB  BldIMG       Write to BldIMG.
2723 1B7B4 68BE  CkMltj GOTO   CkMult
2724              *
```

```
2725                    EJECT
2726                ********************************************************************
2727                ********************************************************************
2728                **
2729                ** Name:    IM"E"   -  Process "E" symbol in IMAGE string
2730                **
2731                ** Category:   LOCAL
2732                ** Purpose:    Process E symbol in numeric IMAGE field.
2733                ** Exit:       To CkDlim.
2734                ** Calls:      WR#DIG, BldIM+
2735                **
2736                ** Algorithm:
2737                **   IM"E"  : Re-initialize delimiter to specify that
2738                **         the numeric field includes an exponent.  Also
2739                **         write out the final digit count.  (WR#DIG)
2740                **       Write out "SZZZ" symbols to BldIMG for outputting
2741                **         exponent sign and digits.
2742                **       Exit to CkDlm+.
2743                **
2744                ** History:
2745                **
2746                **    Date     Programmer              Modification
2747                **  --------   ----------   ---------------------------------
2748                **  12/08/82   MB           Wrote routine, documented.
2749                **
2750                ********************************************************************
2751                ********************************************************************
2752                *
2753 1B7B8           IM"E"                        "E" specifier: exponent.
2754                *                          Note: if sInit=0 (e.g., ".E"), this
2755                *                          will be trapped at CkDlm+.
2756 1B7B8 32CF             LC(3)  uNUMEn          (Extra 0 for WR#DIG.)
          0
2757 1B7BD 7CD1           GOSUB  WR#DIG          Back to uDELIM, write #digits.
2758                *
2759 1B7C1 37A5           LCASC  \SZZZ\          Fill in default "Eszzz" for
           A5A5
           35
2760 1B7CB 27             P=     7                 exponent field.
2761 1B7CD 7992           GOSUB  BldIM+          Write to BldIMG.
2762 1B7D1 842            ST=0   sCntg           No more counting digits.
2763 1B7D4 6EBD  CkDl+j   GOTO   CkDlm+
2764                *
```

```
2765                    EJECT
2766                ********************************************************************
2767                ********************************************************************
2768                **
2769                ** Name:    IM"X"   -  Process "X" symbol in IMAGE string
2770                **
2771                ** Category:   LOCAL
2772                ** Purpose:    Process X symbol in IMAGE string.
2773                ** Entry:      From CkEdit
2774                ** Exit:       To CkMult
2775                ** Calls:      IMmult
2776                **
2777                ** Algorithm:
2778                **    If multiplier pending, write out required fields.
2779                **      In any case, write out X symbol to BldIMG.
2780                **    Exit to CkMult.
2781                **
2782                ** History:
2783                **
2784                **    Date     Programmer             Modification
2785                **  --------   ----------   ----------------------------------
2786                **  12/08/82   MB           Wrote routine, documented.
2787                **
2788                ********************************************************************
2789                ********************************************************************
2790                *
2791 1B7D8 7861   IM"X"  GOSUB  IMmult         Write out to BldIMG.
2792 1B7DC 57D           GONC   CkMltj         (BET) Check for more <edit> chars.
2793                *
```

```
2794                    EJECT
2795              ****************************************************************
2796              ****************************************************************
2797              **
2798              ** Name:    IMsep   -  Process separator symbols in IMAGE
2799              **
2800              ** Category:   LOCAL
2801              ** Purpose:    Process C and P symbols in IMAGE string.
2802              ** Entry:      From CkCPSM
2803              ** Exit:       If C is separator symbol, exit to CkMult.
2804              **             If C is part of C( symbol, exit through CkDlim.
2805              **
2806              ** Algorithm:
2807              **   IM"C":  Move DO to next IMAGE string char.
2808              **           Read next char; if "(", abort subroutine and
2809              **             jump to CkDlim to process the new delimiter.
2810              **   IM"P":  If C or P pending (sC/P=1), then error.
2811              **           Set sC/P=1 ("C or P pending")
2812              **           If multiplier pending, error. Otherwise, write
2813              **             symbol to BldIMG. (NoMult)
2814              **           Exit to CkMult.
2815              **
2816              ** History:
2817              **
2818              **    Date     Programmer             Modification
2819              **  --------   ----------   ----------------------------------
2820              **  12/08/82   MB           Wrote routine, documented.
2821              **
2822              ****************************************************************
2823              ****************************************************************
2824              *
2825 1B7DF         IMsep
2826 1B7DF 871           ?ST=1  sC/P          C/P pending?
2827 1B7E2 75            GOYES  IMerre        Yes.
2828 1B7E4 7171          GOSUB  cNoMlt        Write to BldIMG.
2829 1B7E8 5BC           GONC   CkMltj        (BET)
2830              *
```

```
2831                    EJECT
2832              ************************************
2833              *
2834 1B7EB 316F  IMHKB^ LC(2)  uHKB^          H,K,B or ^ symbol.
2835 1B7EF 7F90         GOSUB  IMint1        Replace delimiter with uHKB^ ,
2836              *                            initialize, check for cmplx.
2837 1B7F3 7561         GOSUB  NoMult        Write to BldIMG, check mult.
2838 1B7F7 5CD          GONC   CkDl+j        (BET) Back to check for delimiter.
2839              *
```

```
2840                    EJECT
2841              ********************************************************
2842              *
2843 1B7FA 845    IM"}"  ST=0   sSTOP       Don't stop for error at open parens.
2844 1B7FD        IM"}"1
2845 1B7FD 7F51          GOSUB  IM","          Error on illegal multiplier.
2846 1B801 7732          GOSUB  D12ROA         Save D1 in R0(A).
2847 1B805 1C1           D1=D1- 2              Start with present char.
2848              *
2849 1B808 7342   BACK2( GOSUB  BACK           Backwards search to left paren.
2850              *
2851 1B80C ED            CON(2) uIMsta           If Unmatched Parentheses:
2852 1B80E 620           REL(3) IMstrt         "IMAGE End": "IMAGE Parentheses."
2853              *
2854 1B811 0E            CON(2) uOPNWM         "Open loop with multiplier."
2855 1B813 850           REL(3) BOPNWM           Stops search.
2856              *
2857 1B816 FD            CON(2) uOPNM-         "Open loop with decremented mult".
2858 1B818 C40           REL(3) BOPNM-           Stops search.
2859              *
2860 1B81B 8D            CON(2) uOPNNM         "Open loop w/o multiplier."
2861 1B81D 530           REL(3) BOPNNM           Stops search.
2862              *
2863 1B820 CD            CON(2) uIMbck         "Poll for backwd search handler."
2864 1B822 851           REL(3) IMGbck           Stops search.
2865              *
2866 1B825 9D            CON(2) uJMP{}         "Jump over loop pointer."
2867 1B827 610           REL(3) BJMP{}           Does not stop search.
2868              *
2869 1B82A AD            CON(2) uJMPst         "Jump over string pointer."
2870 1B82C 020           REL(3) BJMPst           Does not stop search.
2871              *
2872 1B82F 00            CON(2) 0              All other tokens simply
2873 1B831 56D           GONC   BACK2(           jump a byte.
2874              *
```

```
2875                    EJECT
2876                ************************************
2877                *
2878                *
2879 1B834 875    IMstrt ?ST=1  sSTOP
2880 1B837 00             RTNYES
2881 1B839 6F41   IMerre  GOTO   IMerr
2882                *
2883                *
2884                ************************************
2885                *
2886 1B83D 171    BJMP{} D1=D1+ 2
2887 1B840 8E00           GOSUBL =C+A2D1
           00
2888 1B846 135            D1=C
2889 1B849 1C3            D1=D1- 4              D1 back to uMULT token.
2890                *
2891 1B84C 17B    BJMPst D1=D1+ 12
2892 1B84F 58B           GONC   BACK2(
2893                *
2894                ************************************
2895                *
2896 1B852         BOPNNM
2897                *
2898 1B852 875            ?ST=1  sSTOP          Error on open parentheses?
2899 1B855 4E             GOYES  IMerre         Yes.
2900                *
2901 1B857 D2             C=0    A              No. Write a NOP.
2902 1B859 14D            DAT1=C B
2903 1B85C 118    =EndBck C=R0                  Restore D1.
2904 1B85F 137            CD1EX
2905 1B862 03             RTNCC
2906                *
```

```
2907                    EJECT
2908            ****************************************************************
2909            ****************************************************************
2910            **
2911            ** Name:(S) BOPNM-  -  Process uOPNM- token during backup
2912            **
2913            ** Category:   EXCUTL
2914            **
2915            ** Purpose:
2916            **      To process uOPNM- token during IMAGE parse backward
2917            **      search.
2918            **
2919            ** Entry:
2920            **      P      = 0
2921            **      D1 points to uOPNM- token in BldIMG stream
2922            **      R0(A)=current position in BldIMG stream (any new
2923            **        token will be written below this address)
2924            **      R1(A)=address if symbol which caused backward search:
2925            **        a right parenthesis (to close a field), or the end-
2926            **        of-image (to check for unmatched parentheses).
2927            **      S5=1 if end-of-image search; S5=0 if closing field.
2928            **
2929            ** Exit:
2930            **      P      = 0
2931            **      Carry clear.
2932            **      A uLOOPP token, a 5-nibble offset pointing to the left
2933            **        parenthesis location, and a uJMP{} token will have
2934            **        been written to the BldIMG stream.
2935            **      D1=current position in BldIMG stream (address passed
2936            **        in R0(A) minus 9)
2937            **
2938            ** Calls:      COPYm1, EndBck, IMoffs, BldIMG
2939            **
2940            ** Uses.......
2941            **  Exclusive: B(A),C,D1,P
2942            **  Inclusive: B(A),C,D1,P,S8
2943            **
2944            ** Stk lvls:   1
2945            **
2946            ** NOTE:
2947            **      This backward search during IMAGE parsing is performed
2948            **      to find an open field (a field defined by parentheses).
2949            **      The search is perforne either to close the field (when
2950            **      a right parenthesis is found), or to check for
2951            **      unmatched parentheses at the end-of-image.
2952            **
2953            ** Algorithm:
2954            **      Set B(A)=1 (for COPYm1)
2955            **      If S5=1 ("end-of-image"), report "Invalid IMAGE" error.
2956            **      Copy multiplier from reserve field to decrementer
2957            **        field (adding one to the reserve, from B(A))
2958            **      Write uLOOPP token to BldIMG
2959            **      Compute offset to left parenthesis position, store it
2960            **        in BldIMG
2961            **      Write uJMP{} token to BldIMG.
```

```
2962            **
2963            ** History:
2964            **
2965            **    Date      Programmer            Modification
2966            **  --------    ----------   ----------------------------------
2967            **  12/08/82    MB           Documentation
2968            **
2969            ********************************************************************
2970            ********************************************************************
2971            *
2972 1B864 D1  =BOPNM- B=0     A
2973 1B866 E5          B=B+1   A               Increment multiplier.
2974 1B868 540         GONC    BOPWO1          (BET)
2975            *
2976 1B86B D1   BOPNWM B=0     A               No change to multiplier.
2977 1B86D     BOPWO1
2978            *
2979 1B86D 875         ?ST=1   sSTOP           Error on open parentheses?
2980 1B870 9C          GOYES   IMerre          Yes.
2981            *
2982 1B872 1C3         D1=D1-  4               No. Back to multiplier storage.
2983 1B875 7161        GOSUB   COPYm1          Copy multiplier to loop counter.
2984 1B879 7FDF        GOSUB   EndBck          End backwards search.
2985 1B87D 25  =BOPWO5 P=      5
2986 1B87F 31FE        LC(2)   uLOOPP          "Loop on parentheses."
2987 1B883 71D1        GOSUB   IMoffs          Offset to BldIMG.
2988 1B887 319D        LC(2)   uJMP{}          "Jump over parentheses."
2989 1B88B 6CD1        GOTO    BldIMG
2990            *
```

```
2991                    EJECT
2992            ****************************************************************
2993            ****************************************************************
2994            **
2995            ** Name:(S) IMinit  -  Initiate IMAGE output field
2996            ** Name:    IMin01  -  Backup to field delimiter (close field)
2997            **
2998            ** Category:   EXCUTL
2999            **
3000            ** Purpose:
3001            **      To back up through the BldIMG token stream to the
3002            **      pending delimiter and re-write a field delimiter, in
3003            **      order to identify the type of field for the execution
3004            **      routines.
3005            **
3006            ** Entry:
3007            **      P      =  0
3008            **      C(B)=new delimiter token (see detail, below)
3009            **      D1=current position in BldIMG stream (any new tokens
3010            **        will be written below this address)
3011            **      A(B)=IMAGE symbol which caused the initialization (in
3012            **        uppercase)
3013            **      D(A)=AvMemend
3014            **      S3=0 if field has not already been initialized;
3015            **        S3=1 if field has already been initialized.
3016            **
3017            ** Exit:
3018            **      If pending fields need to be executed (S0=1), then
3019            **        exits to IMGxqt.
3020            **      Else,
3021            **          P=0
3022            **          Carry clear
3023            **          D1=current position in BldIMG stream
3024            **          C=address of delimiter token
3025            **          Delimiter token has been re-written to identify
3026            **            new field.
3027            **
3028            ** Calls:       D12R0A, BACK, CSL9R0
3029            **              FPOLL (pIMcpi) if S7=1
3030            **              IMGxqt if S0=1.
3031            **
3032            ** Uses.......
3033            **  Exclusive: A,B(A),C,R0(A),R2
3034            **              IMinit also uses S0,S2,S3,S10
3035            **  Inclusive: If S0=0: A,B,C,R0(A),R2
3036            **              If S0=1: can use anything in execution routines.
3037            **
3038            ** Stk lvls:   3 (unless S0=1: execution routines can use 7)
3039            **
3040            ** NOTE:
3041            **      Whenever a new field begins, a delimiter token
3042            **      (uDELIM) is written to the BldIMG stream, along with
3043            **      two 4-nibble fields used for digit counters.  Also,
3044            **      S3 is set=0 to indicate that the field has not yet been
3045            **      initialized (type of field not yet discovered).  IMinit
```

```
3046            **      is called whenever an output character is found; if
3047            **      S3=1, it returns immediately.  Otherwise, S3 is set=1,
3048            **      and the BldIMG tokens are scanned (backwards) until
3049            **      the uDELIM token is found.  It is then replaced with
3050            **      the appropriate token to identify the type of field.
3051            **
3052            **      However, if pending fields need to be executed (S0=1),
3053            **      the token is replaced with a uRESTP (restart parse)
3054            **      token, and IMGxqt is invoked to execute the fields.
3055            **
3056            **      IMin01 is called to find the field delimiter at certain
3057            **      times, for the following actions:
3058            **        1) when a radix symbol (. or R) is found, one of the
3059            **           4-nibble counter fields is filled with the number
3060            **           of digits before the radix
3061            **        2) when a numeric field ends, the other 4-nibble
3062            **           counter field is filled with the total number of
3063            **           digit symbols.
3064            **        3) when a sign symbol (S or M) is found, the field
3065            **           delimiter is adjusted to indicate that a sign
3066            **           is specified.
3067            **        4) when the E symbol is found, the field delimiter
3068            **           is replaced with one which indicates that the
3069            **           exponent is to be displayed.
3070            **      At these times, S0=0 so that execution will not start.
3071            **
3072            **      Fast poll for pIMcpi may change S0, or the flag in
3073            **      R2(XS) (see C(XS) detail below), if necessary.
3074            **
3075            ** Detail:
3076            **      At entry to IMin01, C(XS) is used as a flag to indicate
3077            **      whether to re-write the delimiter.  In cases (1) and
3078            **      (2) above, the field delimiter is not overwritten; in
3079            **      these cases, C(XS) is nonzero as a flag.
3080            **
3081            **      At entry to IMin01, C(B)=new delimiter token to re-
3082            **      write, or C(B)=0 if delimiter merely has to be ad-
3083            **      justed (case (3) above).
3084            **
3085            ** Algorithm:
3086            **   IMinit: Set S2=1 ("count digits")
3087            **           If S3=1, return. ("Field already initialized")
3088            **           Set S3=1, S0=1 ("execute pending fields"),
3089            **             S10=1 ("output field found")
3090            **           Set C(XS)=0 (flag for "re-write delimiter")
3091            **           Save symbol in R2.
3092            **   IMin01: Save D1 in R0.
3093            **           If S7=1 and S0=1, fast poll (pIMcpi)
3094            **      1)   Back up through tokens:
3095            **             if uJMPst, then D1+12, go to 1)
3096            **             if uJMPdl, then D1+6, go to 1)
3097            **             if uDELIM, then go to 3)
3098            **             if other delimiter, go to 4)
3099            **             if uRESTP, then go to 2)
3100            **             else go to 1)
```

```
3101            **      2)   Set S0=1 (don't execute)
3102            **           Copy D1 to R0(9-5) (new execution address)
3103            **      3)   Clear R2(A) (digit count)
3104            **           If S0=1, jump to IMGxqt: re-write delimiter
3105            **             with uRESPT token and execute pending fields.
3106            **           If "don't re-write delimiter", go to 5)
3107            **           If "write new token", go to 4)
3108            **           If S9=1 ("sign"), then increment delim+1
3109            **      4)   Re-write delimiter
3110            **      5)   Restore D1 from R0(A).
3111            **
3112            ** History:
3113            **
3114            **    Date     Programmer            Modification
3115            **  --------   ----------   ------------------------------------
3116            **  12/08/82   MB           Documentation
3117            **
3118            ****************************************************************
3119            ****************************************************************
3120            *
3121 1B88F 852  =IMinit ST=1   sCntg          "Counting digits."
3122 1B892 873   IMint1 ?ST=1  sInit          Already initialized?
3123 1B895 00           RTNYES               Yes.
3124 1B897 853          ST=1   sInit          No. Set flag.
3125 1B89A 85A          ST=1   sFOUND         "Display field found."
3126 1B89D 850          ST=1   sXQT           New field: "execute IMAGE".
3127 1B8A0 AA2          C=0    XS             Flag: "Re-write delim token."
3128 1B8A3 102          R2=A                  Save char symbol.
3129            *
3130 1B8A6 D5    IMin01 B=C    A              Save token & flag in B.
3131 1B8A8 7091         GOSUB  D12R0A         Save D1 in R0(A).
3132 1B8AC 867          ?ST=0  sCplxP         In Complex field?
3133 1B8AF 41           GOYES  IMin03         No.
3134 1B8B1 860          ?ST=0  sXQT           Yes. Init'lzg or filling count?
3135 1B8B4 F0           GOYES  IMin03         Filling count; no handling.
3136 1B8B6 10A          R2=C                  Token&flag in R2, instead of char.
3137 1B8B9 7AD0         GOSUB  FPOLLj         "Poll for cmplx fld init'lzn."
3138 1B8BD 00           CON(2) =pIMcpi
3139 1B8BF 73C0         GOSUB  Polrtn         Check for handled.
3140            *         +-----------------------------------------------+
3141            *         |  If poll handled, the handler must return:    |
3142            *         |       B(X)= token and flag (from R2(X))       |
3143            *         |       D1 restored (from R0(A))                |
3144            *         |       D(A)= AvMemSt                           |
3145            *         |       R2(B)= Image string character (converted to |
3146            *         |          upper case!).  The character is located  |
3147            *         |          at the address found in R1(A).       |
3148            *         +-----------------------------------------------+
3149 1B8C3 1CD    IMin03 D1=D1- 14             Start with present char (D1-2).
3150 1B8C6 175    BD1+12 D1=D1+ 6
3151 1B8C9 175    BD1+6  D1=D1+ 6
3152            *
3153 1B8CC 7F71   BACK2d GOSUB  BACK           Backwards search to delimiter.
3154            *
3155 1B8D0 AD           CON(2) uJMPst         "Jump over string pointer."
```

```
3156 1B8D2 4FF          REL(3) BD1+12          Does not stop search.
3157            *
3158 1B8D5 BD           CON(2) uJMPd1        "Jump over unfilled delimiter."
3159 1B8D7 2FF          REL(3) BD1+6           Does not stop search.
3160            *
3161 1B8DA 4F           CON(2) uDELIM        This would be covered by test
3162 1B8DC A10          REL(3) BuDLIM        below, but included here as
3163            *                            a service to poll handlers.
3164            *
3165 1B8DF 00           CON(2) 0             Other tokens.
3166 1B8E1 311F         LC(2)  uRESTP        "Restart parse" token.
3167 1B8E5 9E2          ?A<C   B             Keep back-searching?
3168 1B8E8 4E           GOYES  BACK2d        Yes.
3169 1B8EA 9E6          ?A>C   B             No. Found a delim?
3170 1B8ED E0           GOYES  BDELIM        Yes.
3171            *
3172 1B8EF 840          ST=0   sXQT          No. Restart parsing.
3173 1B8F2 7711         GOSUB  CSL9R0        Save D1=xqt addr in R0(9-5).
3174 1B8F6 D0    BuDLIM A=0    A             To clear out digit count in R2.
3175 1B8F8 122          AR2EX                Restore char symbol.
3176            *
3177 1B8FB 860   BDELIM ?ST=0  sXQT          Execute IMAGE?
3178 1B8FE 60           GOYES  BDLIM3        No.
3179 1B900 6000         GOTO   =IMGxqt       Execute.
3180            *
3181 1B904 92D   BDLIM3 ?B#0   XS          From WR#DIG? (fill in #digits only)
3182 1B907 61           GOYES  BDLIM9          Yes.
3183 1B909 D9           C=B    A               No. Delim token back to C.
3184 1B90B 96E          ?C#0   B               Write out new token?
3185 1B90E 50           GOYES  BDLIM5          Yes.
3186 1B910 14F          C=DAT1 B               No. Get old token.
3187 1B913 869   BDLIM5 ?ST=0  sSIGN           Sign specified?
3188 1B916 40           GOYES  BDLIM7          No.
3189 1B918 E6           C=C+1  A               Yes. Change old token.
3190 1B91A 14D   BDLIM7 DAT1=C B               Re-write token.
3191 1B91D 6E3F  BDLIM9 GOTO   EndBck         End backward search.
3192            *
```

```
3193                    EJECT
3194                *****************************************************
3195                *
3196 1B921 841     IMmlt+ ST=0   sC/P          Clear "C/P pending" flag.
3197 1B924 D2              C=0    A
3198 1B926 E6              C=C+1  A            Add one to digit count.
3199 1B928 868             ?ST=0  sMULT        Unless multiplier...
3200 1B92B 50              GOYES  IMmlt0
3201 1B92D 147             C=DAT1 A             .. then add mult.
3202 1B930 122     IMmlt0  AR2EX
3203 1B933 23              P=     3
3204 1B935 05              SETDEC
3205 1B937 A1A             A=A+C  WP           Increase digit count.
3206 1B93A 04              SETHEX
3207 1B93C 20              P=     0
3208 1B93E 4A4             GOC    IMerr        Overflow.
3209                *
3210 1B941 122             AR2EX
3211                *
3212 1B944 868     IMmult  ?ST=0  sMULT        Multiplier pending?
3213 1B947 51              GOYES  BlIMAj       No. Send char to BldIMG.
3214 1B949 7B80            GOSUB  COPYmu       Copy multiplier.
3215 1B94D 7511            GOSUB  BldIMA       Send char to BldIMG.
3216 1B951 312D            LC(2)  uLOOPB       Loop on byte.
3217 1B955 6211            GOTO   BldIMG       Store "loop on byte" in BldIMG.
3218                *
3219                *
3220                *******************************
3221                *
3222 1B959 851     cNoMlt  ST=1   sC/P
3223 1B95C         NoMult
3224 1B95C 7601    BlIMAj  GOSUB  BldIMA       Write symbol to BldIMG.
3225 1B960 878     IM","   ?ST=1  sMULT        Multiplier pending?
3226 1B963 62              GOYES  IMerr        Yes.
3227 1B965 03              RTNCC
3228                *
3229                *
```

```
3230                    EJECT
3231                ********************************************************************
3232                ********************************************************************
3233                **
3234                ** Name:(S) IMerr   -  Report "Invalid IMAGE" error
3235                **
3236                ** Category:   SYSTEM
3237                **
3238                ** Purpose:
3239                **      To generate the error "Invalid IMAGE".
3240                **
3241                ** Entry:
3242                **      No necassary conditions.
3243                **
3244                ** Exit:
3245                **      Through MFERR.
3246                **
3247                ** Calls:      MFERR
3248                **
3249                ** Uses:       MFERR exits to BASIC main loop; may use anything
3250                **
3251                ** Stk lvls:   MFERR exits to BASIC main loop; may use 7
3252                **
3253                ** Detail:
3254                **       =IMerr  P=      0
3255                **               LC(2)   =eINVIM
3256                **               GOVLNG  =MFERR
3257                **
3258                ** History:
3259                **
3260                **    Date     Programmer              Modification
3261                **  --------   ----------   ----------------------------------
3262                **  12/08/82   MB           Documentation
3263                **
3264                ********************************************************************
3265                ********************************************************************
3266                *
3267 1B967          IMGend                          End of IMAGE string.
3268 1B967 7BF0             GOSUB  BldIMA          Write uIMend to BldIMG.
3269 1B96B 72B0             GOSUB  IMD0-2          Don't want D0 incremented past.
3270 1B96F 855              ST=1   sSTOP           For IM"}" .
3271 1B972 778E             GOSUB  IM"}"1          Check back for open parentheses.
3272 1B976 6000             GOTO   =IMGxq1
3273                *
3274                ************************************
3275                *
3276                *
3277 1B97A 137    IMGbck CD1EX                    Pass D1 to handler in R2.
3278 1B97D 10A              R2=C
3279 1B980 7310             GOSUB  FPOLLj
3280 1B984 00               CON(2) =pIMbck
3281                *    +-------------------------------------------+
3282                *    |     If pIMbck handled, handler must return:  |
3283                *    |         D1 restored (from R0(A) -- this is   |
3284                *    |           the D1 from start of bckwd search) |
```

```
3285            *    |         D(A)= AvMemSt                          |
3286            *    +----------------------------------------------+
3287            *
3288 1B986 400  =Polrtn RTNC                    Note: for fast poll XM=0 is
3289            *                                  equivalent to Carry set.
3290 1B989 20   =IMerr  P=     0
3291 1B98B 3100         LC(2)  =eINVIM          "Invalid IMAGE".
3292 1B98F 6E5B         GOTO   kMFERR
3293            *
3294 1B993 75A0 =FPOLLx GOSUB  D12ROA           Pass D1 to handler in R0(A).
3295 1B997 8C00 =FPOLLj GOLONG =fpoll
           00
3296            *
```

```
3297                    EJECT
3298              ************************************
3299              *
3300 1B99D        WR#DIG                          Enter with C(B)= new delim token.
3301 1B99D 7BBF          GOSUB  NoMult            Write symbol to BldIMG.
3302 1B9A1 D6            C=A    A                 C= new delim token.
3303 1B9A3        WR#DG3
3304 1B9A3 871           ?ST=1  sC/P              C/P pending?
3305 1B9A6 3E            GOYES  IMerr             Yes.
3306 1B9A8 7AFE          GOSUB  IMinO1           Back to delim token.
3307 1B9AC 137           CD1EX                   D1= addr of delim.
3308 1B9AF 112           A=R2                    Fetch digit counter.
3309 1B9B2 1C7           D1=D1- 8                To total digit# storage.
3310 1B9B5 1593          DAT1=A 4                Store total #digits.
3311 1B9B9 87B           ?ST=1  sRDX             Counting radix digits?
3312 1B9BC 90            GOYES  WR#DG5           Yes.
3313 1B9BE 173           D1=D1+ 4                To #digits-till-rdx storage.
3314 1B9C1 1593          DAT1=A 4                Write out #digits.
3315 1B9C5 137    WR#DG5 CD1EX                   Restore D1.
3316 1B9C8 03            RTNCC
3317              *
```

```
3318                    EJECT
3319              ********************************************************
3320              *
3321 1B9CA 846   =CLOST+ ST=0   sSpecl
3322 1B9CD D0    =CLOST  A=0    A              Clear 0 nibble of status bits.
3323 1B9CF 09    =SET-ST C=ST                  Set 0 nib of st bits to A(0).
3324 1B9D1 A86           C=A    P
3325 1B9D4 0B            CSTEX                 Leaves C(X)= original status.
3326 1B9D6 03            RTNCC
3327              *
3328              ********************************************************
3329              *
3330 1B9D8 D1     COPYmu B=0    A
3331 1B9DA 147    COPYm1 C=DAT1 A
3332 1B9DD C9            C=B+C  A
3333 1B9DF 173           D1=D1+ 4
3334 1B9E2 15D3          DAT1=C 4
3335 1B9E6 1C3           D1=D1- 4
3336 1B9E9 848    COPYm3 ST=0   sMULT
3337 1B9EC 03            RTNCC
3338              *
3339              ********************************************************
3340              *
3341 1B9EE 878    IM"("  ?ST=1  sMULT          Multiplier pending?
3342 1B9F1 8F            GOYES  COPYm3         Yes. Leave multiplier open.
3343 1B9F3 318D          LC(2)  uOPNNM         No. "Open loop, no multiplier."
3344 1B9F7 507           GONC   BldIMG         (BET) Write to BldIMG.
3345              *
```

```
3346                    EJECT
3347            ********************************************************************
3348            ********************************************************************
3349            **
3350            ** Name:    SetAVM  -  Set D(A)=AvMemSt, D1=AvMemEnd
3351            **
3352            ** Category:   PTRUTL
3353            **
3354            ** Purpose:
3355            **      To set D(A)=AvMemSt, D1=AvMemEnd
3356            **
3357            ** Entry:
3358            **      No necessary conditions
3359            **
3360            ** Exit:
3361            **      Carry clear
3362            **      D(A)=AvMemSt
3363            **      D1=C(A)=AvMemEnd
3364            **
3365            ** Calls:      D=AVMS, D1=AVE
3366            **
3367            ** Uses.......
3368            **  Exclusive: Nothing....
3369            **  Inclusive: C(A), D(A), D1
3370            **
3371            ** Stk lvls:   1
3372            **
3373            ** Detail:
3374            **     =SetAVM GOSUBL =D=AVMS
3375            **             GOLONG =D1=AVE
3376            **
3377            ** History:
3378            **
3379            **    Date     Programmer              Modification
3380            **  --------   ----------   ----------------------------------
3381            **  12/08/82   MB           Documentation
3382            **
3383            ********************************************************************
3384            ********************************************************************
3385 1B9FA 8E00 =SetAVM GOSUBL =D=AVMS
           00
3386            *
3387 1BA00 8C00 =SetAVE GOLONG =D1=AVE
           00
3388            *
```

```
3389                    EJECT
3390                ********************************************************************
3391                ********************************************************************
3392                **
3393                ** Name:(S) CSL9RO  -  Copy D1 to RO(9-5)
3394                **
3395                ** Category:   GENUTL
3396                **
3397                ** Purpose:
3398                **      Copy D1 to RO(9-5) without disturbing the rest of RO.
3399                **
3400                ** Entry:
3401                **      No necessary conditions.
3402                **
3403                ** Exit:
3404                **      P       =  0
3405                **      Carry clear
3406                **
3407                ** Calls:      CSLWP9
3408                **
3409                ** Uses.......
3410                **  Exclusive: A,C(A)
3411                **  Inclusive: A,C(A),P
3412                **
3413                ** Stk lvls:   1
3414                **
3415                ** Detail:
3416                **      =CSL9RO A=RO
3417                **              CD1EX
3418                **              D1=C
3419                **              GOSBVL =CSLWP9
3420                **              C=A     A
3421                **              RO=C
3422                **              RTN
3423                **
3424                ** History:
3425                **
3426                **    Date      Programmer              Modification
3427                **  --------    ----------   ------------------------------------
3428                **  12/08/82    MB           Documentation
3429                **
3430                ********************************************************************
3431                ********************************************************************
3432                *
3433 1BA06 8D00    =CSLW9j GOVLNG =CSLWP9
           000
3434                *
3435 1BA0D 110     =CSL9RO A=RO
3436 1BA10 137      CSL9R+ CD1EX
3437 1BA13 135             D1=C
3438 1BA16 7CEF    =CSL2RO GOSUB  CSLW9j
3439 1BA1A D6              C=A     A
3440 1BA1C 108             RO=C
3441 1BA1F 01              RTN
3442                *
```

```
3443                    EJECT
3444            ****************************************************************
3445            ****************************************************************
3446            **
3447            ** Name:(S) IMDO+2  -  Add 2 to R1(A), copy value to DO
3448            ** Name:(S) IMDO-2  -  Subtract 2 from R1(A)
3449            **
3450            ** Category:   GENUTL
3451            **
3452            ** Purpose:
3453            **      IMDO+2: Take DO storage in R1, increment by 2 and copy
3454            **              to DO.
3455            **      IMDO-2: Subtract 2 from R2(A).
3456            **
3457            ** Entry:
3458            **      No necessary conditions.
3459            **
3460            ** Exit:
3461            **      Carry clear.
3462            **      IMDO+2: R1(A) incremented by 2.
3463            **              DO=C(A)=R1(A)
3464            **      IMDO-2: R1(A) decremented by 2.
3465            **
3466            ** Calls:      none
3467            **
3468            ** Uses.......
3469            **  Exclusive:
3470            **      IMDO+2: C(W), DO
3471            **      IMDO-2: nothing
3472            **
3473            ** Stk lvls:   0
3474            **
3475            ** Detail:
3476            **     =IMDO-2 CR1EX
3477            **             C=C-1  A
3478            **             C=C-1  A
3479            **             CR1EX
3480            **             RTNCC
3481            **     =IMDO+2 C=R2
3482            **             C=C+1  A
3483            **             C=C+1  A
3484            **             R1=C
3485            **             DO=C
3486            **             RTNCC
3487            **
3488            ** History:
3489            **
3490            **    Date     Programmer              Modification
3491            **  --------   ----------   ------------------------------------
3492            **  12/08/82   MB           Documentation
3493            **
3494            ****************************************************************
3495            ****************************************************************
3496            *
3497 1BA21 129  =IMDO-2 CR1EX
```

```
3498 1BA24 CE    =IMDO-- C=C-1  A
3499 1BA26 CE            C=C-1  A
3500 1BA28 129           CR1EX
3501 1BA2B 03            RTNCC
3502             *
3503             ************************************
3504             *
3505 1BA2D 119   =IMDO+2 C=R1
3506 1BA30 E6            C=C+1  A
3507 1BA32 E6            C=C+1  A
3508 1BA34 109           R1=C
3509 1BA37 134           DO=C
3510 1BA3A 03            RTNCC
3511             *
```

```
3512                    EJECT
3513                ****************************************************************
3514                ****************************************************************
3515                **
3516                ** Name:(S) D12ROA  -  Copy D1 to R0(A)
3517                **
3518                ** Category:   GENUTL
3519                **
3520                ** Purpose:
3521                **      To copy D1 to R0(A) without disturbing the rest of R0.
3522                **
3523                ** Entry:
3524                **      No necessary conditions.
3525                **
3526                ** Exit:
3527                **      Carry clear.
3528                **
3529                ** Calls:      none
3530                **
3531                ** Uses.......
3532                **  Exclusive: R0(A)
3533                **
3534                ** Stk lvls:   0
3535                **
3536                ** Detail:
3537                **      =D12ROA CROEX
3538                **              CD1EX
3539                **              D1=C
3540                **              CROEX
3541                **              RTNCC
3542                **
3543                ** History:
3544                **
3545                **    Date      Programmer              Modification
3546                **   --------   ----------   ---------------------------------
3547                **   12/08/82   MB           Documentation
3548                **
3549                ****************************************************************
3550                ****************************************************************
3551 1BA3C 128  =D12ROA CROEX                  D1 to R0(A).
3552 1BA3F 137         CD1EX
3553 1BA42 135         D1=C
3554 1BA45 128         CROEX
3555 1BA48 03          RTNCC
3556                *
```

```
3557                    EJECT
3558                ********************************************************************
3559                ********************************************************************
3560                **
3561                ** Name:    BACK    -  Parse scan of DAT1, increment D1+2
3562                ** Name:    BYTscA  -  Parse scan of DAT1, decrement D1-2
3563                **
3564                ** Category:   EXCUTL
3565                **
3566                ** Purpose:
3567                **      Read a byte from DAT1, scan a table of values for a
3568                **      match.  If match found, jump to corresponding routine.
3569                **
3570                ** Entry:
3571                **      BACK:   P=0
3572                **              Byte for matching is at D1+2.
3573                **      BYTscA: Byte for matching is at D1-2.
3574                **              Table of byte values and relative offsets
3575                **                resides as RSTK address (as for FINDA)
3576                ** Exit:
3577                **      P      =  0
3578                **      Carry clear
3579                **      Exits to desired routine if byte match, returns
3580                **        to address past table if no match.
3581                **
3582                ** Calls:      FINDA
3583                **
3584                ** Uses.......
3585                **  Exclusive: A(B),D1
3586                **  Inclusive: A(C),D1,C(A),P
3587                **
3588                ** Stk lvls:   1
3589                **
3590                ** NOTE:
3591                **      See FINDA for description of table.
3592                **
3593                ** Detail:
3594                **     =BYTscA P=     0
3595                **             D1=D1- 4
3596                **     =BACK   D1=D1+ 2
3597                **             A=DAT1 B
3598                **             GOVLNG =FINDA
3599                **
3600                ** History:
3601                **
3602                **    Date     Programmer             Modification
3603                **  --------   ----------   ----------------------------------
3604                **  12/08/82   MB           Documentation
3605                **
3606                ********************************************************************
3607                ********************************************************************
3608                *
3609 1BA4A 20      =BYTscA P=     0
3610 1BA4C 1C3             D1=D1- 4              (Make next stmt a D1=D1-2)
3611                *
```

```
3612 1BA4F 171  =BACK   D1=D1+ 2            Back to last IMAGE token.
3613 1BA52 14B          A=DAT1 B
3614 1BA55 514          GONC   FINDAj       (BET)
3615            *
```

```
3616                    EJECT
3617            ****************************************************************
3618            ****************************************************************
3619            **
3620            ** Name:(S) BldIMG  -  Put tokens from C into BldIMG stream
3621            ** Name:(S) BldIMA  -  Put 1 or 2 tokens from A into BldIMG
3622            ** Name:(S) BldIM+  -  Put tokens from C into BldIMG stream
3623            **
3624            ** Category:   EXCUTL
3625            **
3626            ** Purpose:
3627            **      To put IMAGE tokens into parse stream.
3628            **
3629            ** Entry:
3630            **      BldIMA:   A(B)=token and P=0
3631            **             or A(3-0)=2 tokens and P=2
3632            **      BldIMG:   C=tokens and P=2*(#tokens-1)
3633            **      BldIM+:   C(WP)=tokens and P=2*(#tokens)-1
3634            **                D1=current position in BldIMG stream
3635            **                D(A)=AvMEmSt
3636            **
3637            ** Exit:
3638            **      P      =  0
3639            **      Carry clear
3640            **      Exits to MEMERR if D1 moves below AvMemSt
3641            **
3642            ** Calls:      none
3643            **
3644            ** Uses.......
3645            **  Exclusive: P,D1 moved below write
3646            **      BldIMA: also does ACEX A
3647            **
3648            ** Stk lvls:   0
3649            **
3650            ** NOTE:
3651            **      The "BldIMG" stream refers to the token stream used
3652            **      for IMAGE execution.  This routine can be used by any
3653            **      code which needs to write bytes or nibbles to Available
3654            **      Memory.
3655            **
3656            **      Exampl: for entry into BldIMG, say C(7-0) contains
3657            **      4 tokens.  Then enter with P=6.
3658            **
3659            ** Detail:
3660            **     =BldIMA ACEX   A
3661            **     =BldIMG P=P+1
3662            **     =BldIM+ C=-C   A
3663            **             C+P+1
3664            **             C=-C   A
3665            **             ?C<=D  A
3666            **             GOYES  MEMERR
3667            **             CD1EX
3668            **             DAT1=C WP
3669            **             P=     0
3670            **             RTNCC
```

```
3671            **
3672            ** History:
3673            **
3674            **    Date     Programmer            Modification
3675            **  --------   ----------   ----------------------------------
3676            **  12/08/82   MB           Documentation
3677            **
3678            ************************************************************************
3679            ************************************************************************
3680            *
3681            ************************************************************************
3682            ************************************************************************
3683            **
3684            ** Name:(S) IMoffs  -  Store offset from D1 in BldIMG stream
3685            **
3686            ** Category:   EXCUTL
3687            **
3688            ** Purpose:
3689            **      Store a 5-nibble offset from D1 in the BldIMG stream.
3690            **
3691            ** Entry:
3692            **      P= at least 4.  If C(15-5) contains more tokens to
3693            **        write into the BldIMG stream, then set P such that
3694            **        a P=P+1 will define the entire write field in C(WP).
3695            **      C(A)=address-2 for which offset will be computed.
3696            **
3697            ** Exit:
3698            **      P      =  0
3699            **      Carry clear
3700            **
3701            ** Calls:      BldIMA
3702            **
3703            ** Uses.......
3704            **  Exclusive: C(A)
3705            **  Inclusive: P,D1 (does not use A)
3706            **
3707            ** Stk lvls:   0
3708            **
3709            ** Detail:
3710            **     =IMoffs AD1EX
3711            **             C=C-A  A
3712            **             AD1EX
3713            **             C=C+1  A
3714            **             C=C+1  A
3715            **             ACEX   A
3716            **           <falls into BldIMA>
3717            **
3718            ** History:
3719            **
3720            **    Date     Programmer            Modification
3721            **  --------   ----------   ----------------------------------
3722            **  12/08/82   MB           Documentation
3723            **
3724            ************************************************************************
3725            ************************************************************************
```

```
3726                *
3727 1BA58 133  =IMoffs AD1EX                   Compute offset to IMAGE.
3728 1BA5B E2          C=C-A  A                 Offset.
3729 1BA5D 133         AD1EX                    Restore D1.
3730 1BA60 E6          C=C+1  A
3731 1BA62 E6          C=C+1  A
3732 1BA64 DE          ACEX   A                 (nullify next ACEX)
3733                *
3734                *
3735 1BA66 DE   =BldIMA ACEX   A                 C(B)= symbol; save C in A.
3736                *                            Next statement sends 2 nibs out.
3737 1BA68 0C   =BldIMG P=P+1
3738 1BA6A 137  =BldIM+ CD1EX
3739 1BA6D FA          C=-C   A
3740 1BA6F 809         C+P+1                    Move D1 down #nibs.
3741 1BA72 FA          C=-C   A
3742 1BA74 8BB         ?C<=D  A                 Past AVMEMS?
3743 1BA77 72          GOYES  jMEMER            Yes. MEM ERR.
3744 1BA79 137         CD1EX
3745 1BA7C 1551        DAT1=C WP                Write out symbols to BldIMG.
3746 1BA80 20          P=     0
3747 1BA82 03          RTNCC
3748                *
```

```
3749                    EJECT
3750            ****************************************************************
3751            ****************************************************************
3752            **
3753            ** Name:(S) PRSscn  -  IMAGE parse scan
3754            ** Name:(S) PRSsc+  -  IMAGE parse scan, increment D0 first
3755            **
3756            ** Category:   EXCUTL
3757            **
3758            ** Purpose:
3759            **      Read a byte from address in R1(A), scan a table of
3760            **      values for a match.  If match found, jump to corres-
3761            **      ponding routine.
3762            **
3763            ** Entry:
3764            **      P      =  0
3765            **      R1(A)=address of byte to match
3766            **      Address in RSTK points to table of bytes and relative
3767            **        offsets (see FINDA for table structure)
3768            **
3769            ** Exit:
3770            **      P      =  0
3771            **      Carry clear
3772            **      Exits to desired routine if byte match.  If no match,
3773            **        returns to address past table.
3774            **
3775            ** Calls:      CONVUC, FINDA
3776            **
3777            ** Uses.......
3778            **  Exclusive: C(W),D0,A(B)
3779            **         PRSsc+ also increments R1(A) by 2.
3780            **  Inclusive: C(W),D0,A(B)
3781            **
3782            ** Stk lvls:   2
3783            **
3784            ** NOTE:
3785            **      The byte from the address found in R1(A) is read into
3786            **      A(B) and converted into upper case before the jump to
3787            **      FINDA.
3788            **
3789            **      See FINDA for description of table of bytes and offsets.
3790            **
3791            ** Detail:
3792            **     =PRSsc+ GOSUB  IMD0+2      Increment R1(A) by 2.
3793            **     =PRSscn C=R1
3794            **             D0=C
3795            **             A=DAT0 B
3796            **             GOSUBL =CONVUC     Convert to upper case.
3797            **             GOVLNG =FINDA
3798            **
3799            ** History:
3800            **
3801            **    Date     Programmer             Modification
3802            **  --------   ----------   ----------------------------------
3803            **  12/08/82   MB           Documentation
```

```
3804            **
3805            ********************************************************************
3806            ********************************************************************
3807            *
3808 1BA84 75AF =PRSsc+ GOSUB  IMDO+2          Increment DO (in R1) by 2.
3809 1BA88 119  =PRSscn C=R1                  Fetch DO, don't advance.
3810 1BA8B 134          DO=C
3811 1BA8E 14A          A=DATO B              Next IMAGE char.
3812 1BA91 8E00         GOSUBL =CONVUC        Convert to upper case.
           00
3813 1BA97 8D00 =FINDAj GOVLNG =FINDA
           000
3814            *
3815            ************************************
3816 1BA9E 8C00  jMEMER GOLONG =MEMERj        GOTO MEMERR
           00
3817            *
3818 1BAA4              END
```

```
 A-chr    Abs     8192 #02000 -   189  2110  2147
=BACK     Abs   113231 #1BA4F -  3612  2849  3153
 BACK2(   Abs   112648 #1B808 -  2849  2873  2892
 BACK2d   Abs   112844 #1B8CC -  3153  3168
 BD1+12   Abs   112838 #1B8C6 -  3150  3156
 BD1+6    Abs   112841 #1B8C9 -  3151  3159
 BDELIM   Abs   112891 #1B8FB -  3177  3170
 BDLIM3   Abs   112900 #1B904 -  3181  3178
 BDLIM5   Abs   112915 #1B913 -  3187  3185
 BDLIM7   Abs   112922 #1B91A -  3190  3188
 BDLIM9   Abs   112925 #1B91D -  3191  3182
 BJMPst   Abs   112716 #1B84C -  2891  2870
 BJMP{}   Abs   112701 #1B83D -  2886  2867
=BOPNM-   Abs   112740 #1B864 -  2972  2858
 BOPNNM   Abs   112722 #1B852 -  2896  2861
 BOPNWM   Abs   112747 #1B86B -  2976  2855
 BOPW01   Abs   112749 #1B86D -  2977  2974
=BOPW05   Abs   112765 #1B87D -  2985
=BYTscA   Abs   113226 #1BA4A -  3609
 BlIMAj   Abs   112988 #1B95C -  3224  3213
=BldIM+   Abs   113258 #1BA6A -  3738  2554  2712  2761
=BldIMA   Abs   113254 #1BA66 -  3735  3215  3224  3268
=BldIMG   Abs   113256 #1BA68 -  3737  1850  1882  2102  2722  2989  3217  3344
 BuDLIM   Abs   112886 #1B8F6 -  3174  3162
 C+A2D1   Ext                 -  2887
 C-chr    Abs       32 #00020 -   197   204
 CKINFO   Ext                 -  1873
=CLOST    Abs   113101 #1B9CD -  3322
=CLOST+   Abs   113098 #1B9CA -  3321  2105
 CONVUC   Ext                 -  3812
 COPYm1   Abs   113114 #1B9DA -  3331  2983
 COPYm3   Abs   113129 #1B9E9 -  3336  3342
 COPYmu   Abs   113112 #1B9D8 -  3330  2719  3214
 CP       Abs       36 #00024 -   204   208
 CPE      Abs      100 #00064 -   208  2190  2197  2236  2273
=CSL2RO   Abs   113174 #1BA16 -  3438
 CSL9R+   Abs   113168 #1BA10 -  3436  1887
=CSL9RO   Abs   113165 #1BA0D -  3435  3173
=CSLW9j   Abs   113158 #1BA06 -  3433  1879  3438
 CSLWP9   Ext                 -  3433
 CSRW9j   Ext                 -  2685  2692
 CkD1+j   Abs   112596 #1B7D4 -  2763  2838
 CkD103   Abs   112048 #1B5B0 -  2059  2057
 CkD107   Abs   112057 #1B5B9 -  2063  2060
 CkD1BS   Abs   111959 #1B557 -  1974  2063
 CkDlim   Abs   112027 #1B59B -  2052  2605
 CkDm+    Abs   112019 #1B593 -  2049  2007  2763
=CkLoop   Abs   112233 #1B669 -  2534  1939  2302
=CkLpNC   Abs   112237 #1B66D -  2538  2049  2109  2349  2385
 CkM103   Abs   112276 #1B694 -  2557  2544
 CkM115   Abs   112298 #1B6AA -  2569  2542
 CkM117   Abs   112331 #1B6CB -  2583  2573  2580
 CkM121   Abs   112352 #1B6E0 -  2591  2601  2604
 CkM125   Abs   112383 #1B6FF -  2603  2594  2598
 CkM127   Abs   112396 #1B70C -  2611  2596
```

```
 CkMltj  Abs  112564 #1B7B4 -  2723  2717  2792  2829
 CkMult  Abs  112237 #1B66D -  2539  2429  2567  2571  2723
 D-chr   Abs   16384 #04000 -   188  2110  2190  2197  2350  2386
=D12ROA  Abs  113212 #1BA3C -  3551  2846  3131  3294
 D1=AVE  Ext                -  3387
 D=AVMS  Ext                -  1875  3385
 DRANGE  Ext                -  2541
 Dblqt   Abs    2048 #00800 -   191   203
 E-chr   Abs      64 #00040 -   196   208  2303  2350
=EndBck  Abs  112732 #1B85C -  2903  2984  3191
=EndNum  Abs     230 #000E6 -    62
 FINDA   Ext                -  3813
=FINDAj  Abs  113303 #1BA97 -  3813  3614
=FPOLLj  Abs  113047 #1B997 -  3295  3137  3279
=FPOLLx  Abs  113043 #1B993 -  3294  1995
 IM"("   Abs  113134 #1B9EE -  3341  1989
 IM"*"   Abs  112144 #1B610 -  2234  2624
 IM","   Abs  112992 #1B960 -  3225  1977  2301  2845
 IM"A"   Abs  112101 #1B5E5 -  2144  2615
 IM"D"   Abs  112113 #1B5F1 -  2185  2614
 IM"E"   Abs  112568 #1B7B8 -  2753  2622
 IM"X"   Abs  112600 #1B7D8 -  2791  2613
 IM"Z"   Abs  112152 #1B618 -  2271  2621
 IM"}"   Abs  112634 #1B7FA -  2843  1986
 IM"}"1  Abs  112637 #1B7FD -  2844  3271
 IM1"Z"  Abs  112160 #1B620 -  2300  2625
 IMCHRp  Abs  111995 #1B57B -  1995  2053
=IMD0+2  Abs  113197 #1BA2D -  3505  1812  2687  3808
=IMD0--  Abs  113188 #1BA24 -  3498
=IMD0-2  Abs  113185 #1BA21 -  3497  1933  3269
 IMDrdx  Abs  112130 #1B602 -  2194  2187
 IMGbck  Abs  113018 #1B97A -  3277  2864
 IMGend  Abs  112999 #1B967 -  3267  1992
 IMGxq1  Ext                -  2010  3272
 IMGxqt  Ext                -  3179
 IMHKB^  Abs  112619 #1B7EB -  2834  2628
 IMLoop  Abs  112229 #1B665 -  2532  2146  2189
 IMLpnu  Abs  112225 #1B661 -  2531  2196  2235  2272
 IMentr  Abs  111925 #1B535 -  1924  1930
=IMerr   Abs  113033 #1B989 -  3290  2608  2881  3208  3226  3305
 IMerrb  Abs  112499 #1B773 -  2698
 IMerrc  Abs  112317 #1B6BD -  2577  2421  2562
 IMerrd  Abs  112392 #1B708 -  2608  2577  2698
 IMerre  Abs  112697 #1B839 -  2881  2827  2899  2980
 IMi"*"  Abs  112144 #1B610 -  2233
 IMi"A"  Abs  112101 #1B5E5 -  2143
 IMi"D"  Abs  112113 #1B5F1 -  2184
 IMi"Z"  Abs  112152 #1B618 -  2270
 IMin01  Abs  112806 #1B8A6 -  3130  2428  3306
 IMin03  Abs  112835 #1B8C3 -  3149  3133  3135
=IMinit  Abs  112783 #1B88F -  3121  2532
 IMint1  Abs  112786 #1B892 -  3122  2835
 IMisgn  Abs  112191 #1B63F -  2384  2426
 IMmlt+  Abs  112929 #1B921 -  3196  2536
 IMmlt0  Abs  112944 #1B930 -  3202  3200
```

```
 IMmult  Abs  112964 #1B944 -  3212  1980  1983  2791
=IMoffs  Abs  113240 #1BA58 -  3727  2690  2987
 IMprct  Abs  111948 #1B54C -  1938  1927
 IMrdx   Abs  112172 #1B62C -  2344  2616  2627
 IMsep   Abs  112607 #1B7DF -  2825  2623  2626
 IMsign  Abs  112199 #1B647 -  2419  2619  2620
 IMstr   Abs  112451 #1B743 -  2677  2617  2618
 IMstr3  Abs  112497 #1B771 -  2697  2705
 IMstr7  Abs  112519 #1B787 -  2709  2702
 IMstrt  Abs  112692 #1B834 -  2879  2852
 IMtbl   Ext                -  2589
=InhEOL  Abs       4 #00004 -   178
=IvUSGj  Abs  111848 #1B4E8 -  1864  1828  1844
 LINSKP  Ext                -  1824
 M-chr   Abs     256 #00100 -   194   205
 MEMERj  Ext                -  3816
 MFERR0  Ext                -  1868
 MOVED0  Ext                -  1853
 Nf      Abs       1 #00001 -   202  2110
 NoMult  Abs  112988 #1B95C -  3223  2424  2837  3301
=Nxtfl3  Abs  112086 #1B5D6 -  2104
 Nxtfld  Abs  112061 #1B5BD -  2095  1934  2005
 P-chr   Abs       4 #00004 -   200   204
 PFINDL  Ext                -  1814
 PRSCKB  Ext                -  1811
=PRSsc+  Abs  113284 #1BA84 -  3808  1924  2539
=PRSscn  Abs  113288 #1BA88 -  3809  1974
=Polrtn  Abs  113030 #1B986 -  3288  1997  3139
 Pt-chr  Abs    4096 #01000 -   190   206
 R-chr   Abs       2 #00002 -   201   206
 R3=D10  Ext                -  1886
 REVPOP  Ext                -  1874
 Rx      Abs    4098 #01002 -   206   207  2386
 S-chr   Abs     512 #00200 -   193   205
=SET-ST  Abs  113103 #1B9CF -  3323
 SM      Abs     768 #00300 -   205   207  2197  2350
=STMTNF  Abs  111846 #1B4E6 -  1861  1815
=SetAVE  Abs  113152 #1BA00 -  3387
=SetAVM  Abs  113146 #1B9FA -  3385  1848
 Sglqt   Abs    1024 #00400 -   192   203
 TBLJMP  Ext                -  2611
 USG003  Abs  111746 #1B482 -  1823  1842
 USG005  Abs  111770 #1B49A -  1832  1821
 USG007  Abs  111814 #1B4C6 -  1848  1808
 USG009  Abs  111861 #1B4F5 -  1872  1852
 USG011  Abs  111888 #1B510 -  1878  1857
=USING   Abs  111686 #1B446 -  1803
 WR#DG3  Abs  113059 #1B9A3 -  3303  2061
 WR#DG5  Abs  113093 #1B9C5 -  3315  3312
 WR#DIG  Abs  113053 #1B99D -  3300  2347  2757
 X-chr   Abs   32768 #08000 -   187   203
 Z-chr   Abs     128 #00080 -   195  2110  2273  2386
 Z1-chr  Abs       8 #00008 -   199  2190  2236
 astrsk  Abs      16 #00010 -   198  2110  2236  2386
 cNoMlt  Abs  112985 #1B959 -  3222  2828
```

```
 eINVIM  Ext                  -  3291
 eINVUS  Ext                  -  1866
 eSTMNF  Ext                  -  1867
 ed      Abs   35840 #08C00   -   203   207  2050  2147  2197  2350  2386
 edSMRx  Abs   40706 #09F02   -   207  2110  2190  2236  2273  2303
 expr    Ext                  -  1872
 fpoll   Ext                  -  3295
 jMEMER  Abs  113310 #1BA9E   -  3816  3743
=kMFERR  Abs  111854 #1B4EE   -  1868  3292
 pIMCHR  Ext                  -  1996
 pIMbck  Ext                  -  3280
 pIMcpi  Ext                  -  3138
=sC/P    Abs       1 #00001   -   174  2826  3196  3222  3304
=sCntg   Abs       2 #00002   -   175  2059  2762  3121
=sCplxP  Abs       7 #00007   -   181  3132
=sFOUND  Abs      10 #0000A   -   168  3125
=sInit   Abs       3 #00003   -   176  1938  2052  2425  2602  3122  3124
=sMULT   Abs       8 #00008   -   166  2543  2545  2572  2581  2716  3199  3212
                                3225  3336  3341
=sRDX    Abs      11 #0000B   -   169  2097  2186  2348  3311
=sSIGN   Abs       9 #00009   -   167  2096  2420  2423  3187
=sSTOP   Abs       5 #00005   -   179  2843  2879  2898  2979  3270
=sSpecl  Abs       6 #00006   -   180  3321
=sXQT    Abs       0 #00000   -   173  3126  3134  3172  3177
 tEOL    Ext                  -  1826
 tLBLST  Ext                  -  1840
 tLINE#  Ext                  -  1806
=uALit   Abs     247 #000F7   -    78  2145
=uCPLXC  Abs     238 #000EE   -    69
=uDELIM  Abs     244 #000F4   -    75  2101  3161
=uHKB^   Abs     246 #000F6   -    77  2834
=uIMXCH  Abs     212 #000D4   -    49
=uIMbck  Abs     220 #000DC   -    56  2863
=uIMend  Abs     240 #000F0   -    71  1849  1991
=uIMsta  Abs     222 #000DE   -    57  1881  2851
=uJMPdl  Abs     219 #000DB   -    55  2054  2099  3158
=uJMPst  Abs     218 #000DA   -    54  2710  2869  3155
=uJMP{}  Abs     217 #000D9   -    53  2866  2988
=uLOOPB  Abs     210 #000D2   -    47  3216
=uLOOPP  Abs     239 #000EF   -    70  2986
=uLOOPS  Abs     211 #000D3   -    48  2721
=uMULT   Abs     209 #000D1   -    46  2551
=uNUMEn  Abs     252 #000FC   -    83    84  2756
=uNUMEs  Abs     253 #000FD   -    84
=uNUMFn  Abs     250 #000FA   -    81    82  2188
=uNUMFs  Abs     251 #000FB   -    82
=uNUMNn  Abs     248 #000F8   -    79    80  2531
=uNUMNs  Abs     249 #000F9   -    80
=uOPNM-  Abs     223 #000DF   -    58    59  2857
=uOPNNM  Abs     216 #000D8   -    52  2860  3343
=uOPNWM  Abs     224 #000E0   -    59  2549  2854
=uRESTP  Abs     241 #000F1   -    74  3166
=uSTRPT  Abs     208 #000D0   -    45    46  2689
```

Input Parameters

  Source file name is MB&IMG::MS

  Listing file name is MB/IMG:TI:ML::-1

  Object file name is MB%IMG:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    M   M  BBBB    &     U   U   SSS    GGG
 2            *    MM MM  B   B  & &    U   U  S   S  G   G
 3            *    M M M  B   B  & &    U   U  S      G
 4            *    M M M  BBBB    &     U   U   SSS   G GGG
 5            *    M   M  B   B  & & &  U   U      S  G   G
 6            *    M   M  B   B  &  &   U   U  S   S  G   G
 7            *    M   M  BBBB    && &   UUU    SSS    GGG
 8
 9                      TITLE  USING Execution Routines <831212.1519>
10 1BAA4               ABS    #1BAA4
11            *
12                      RDSYMB MB%IMG::MS
13           ************************************************************
14           *  FIXED ENTRY POINTS (labels which are not used in the
15           *     HP-71 mainframe, but which should be kept as
16           *     external entry points for applications):
17           *        IMxq27
18           *        USGch-
19           *        USGch+
20           *        USst03
21           *        USst05
22           *        USGrst
23           *        USnm05
24           *        ENDIMG
25           *        NwOFFS
26           *        RCVOFS
27           *        C+A2D1
28           *        GetEXP
29           *        TstEnd
30           *        DCRMNT
31           *        USloop
32           *        NXTEXP
33           *        COUNTC
34           *
```

```
35                EJECT
36              ********************************************************
37              * STATUS BITS:
38              *     Note: STR$SB uses S0 and S1.  This does not cause
39              *       any problems as long as the status bit values
40              *       below are used.
41              *           STR$SB uses S0=DSform
42              *                       S1=Blanks
43              *--------------------------------------------------------
44              *                    Numeric values:
45               s"S,M" EQU    sXQT      0  "S" or "M" specified in numeric fld.
46               sFLOAT EQU    sC/P      1  Numeric field with floating chars.
47               sNUME  EQU    sCntg     2  Numeric field with exponent.
48               sZeros EQU    sInit     3  In the zeros.
49              *
50               sKnotH EQU    sZeros    3  "K" specifier, not "H".
51               sRturn EQU    sSTOP     5  Rtn to call routine after disp char.
52              *sSpecl EQU    sSpecl    6  Outputting NaN/Inf; Ovrflw filling.
53               sCplxW EQU    sCplxP    7  Complex number being output.
54              *
55               sStall EQU    sXQT      0  Stalling: no output.
56               sOvfFl EQU    sC/P      1  Overflow filling (fill with *).
57              *
58              ****************************************
59              *  These status bits from MB&IMG must be preserved
60              *  during execution:
61              *       sMULT
62              *       sSIGN
63              *       sFOUND
64              *
65              ************************************************************
66              *  When execution starts, a save area is set up at the
67              *  bottom (low address) of the stack, below the expanded
68              *  BldIMG string:
69              *
70              *           5 nibs 3 nibs 5 nibs 5 nibs  5 nibs
71              *    ------------------------------------------------
72              *          |      |    |      |      |      | BldIMG...
73              *  / ------------------------------------------------
74              * used for   \   ^    ^      ^     ^       ^
75              * expr only  |   |    |      |     |  length of IMAGE string
76              *      AvMemEnd  |    |      |    offset to DO pointer
77              * offset to xqt addr  |  offset to start of IMAGE string
78              *                  status bits
79              *
80              *
81              *   Available-memory-end is set to the storage location of the
82              *   offset to the execution address, and remains there through
83              *   Expression Execute.
84              *
85              *   Memory below AvMemEnd is used only to store an expression
86              *   (put there by a call to EXPEXC).  After return from
87              *   EXPEXC, the offset to the execution address is replaced
88              *   by the offset to the next character to output (or in the
89              *   case of H,K,B or ^, by the offset to the beginning of the
```

```
90          *   stack item).
91          *
92          *    Just before calling EXPR:              5 nibs
93          *      ------------------------------------------- --
94          *                                        |      |
95          *      ------------------------------------------- --
96          *                                AvMemEnd --+  /
97          *                               offset to xqt addr
98          *
99          *    After returning from EXPR:
100         *      ------------------------------------------
101         *        |                               |      |
102         *      ------------------------------------------
103         *                        /       AvMemEnd --+ /
104         *    expr put onto stack        offset into expr
105         *
106         ****************************************************************
```

```
                EJECT
*****************************************************************
* Scratch Register usage:
*
*   R0(A)   = execution address (maintained during external
*             subroutine calls).
*   R0(9-5) = counter for #zeroes in numeric field (maintained
*             during call to SENDIT).
*   R0(S)   = flag to identify numeric symbol:
*              0= *
*              1= Z
*              5= D
*
*   R1(A)   = temporary storage of zero counter (see USGnum).
*             or temporary storage of xqt address while
*             counting floating positions.
*
*   R3(A)   = Program Counter (maintained during external
*             subroutine calls).
*          During float check,
*              R3(A)=counter of floats
*              R3(9-5)=undecremented float counter
*              R3(14-10)=PC
*
*
*   At entry, scratch registers are as described in MB&IMG;
*   when exiting through USGrst ("re-start parse"), they
*   have been restored for MB&IMG.
*
*****************************************************************
```

```
137                 EJECT
138             ********************************************************************
139             ********************************************************************
140             **
141             ** Name:(S) pIMXQT  -  Begin IMAGE execution
142             **
143             ** Category:   POLL
144             **
145             ** Type:       FPOLL
146             **
147             ** Purpose:
148             **      To alert LEX files that IMAGE fields are about to
149             **      begin executing.
150             **
151             ** Should poll be "Handled" (return with XM=0)?:
152             **      No. The IMAGE routines do not check if poll handled.
153             **
154             ** Meaning of "Handling" Poll (what does code do if handled?):
155             **      None.
156             **
157             ** Entry conditions for handler (registers, ST, RAM, etc.):
158             **      Carry set.
159             **      B[A] = Poll number.
160             **      HEX mode.
161             **      P=0.
162             **      R0(9-5)=address of token in BldIMG stream where
163             **         execution is to start.
164             **      R1(A)=address of last character to be parsed in
165             **         IMAGE string.
166             **      R3(A)=Program Counter (original DO as passed to
167             **         the USING routine, updated as expressions are
168             **         executed).
169             **      RAM usage as shown below, in NOTE.
170             **
171             ** Normal exit conditions from handler if handled (ST, RAM,
172             ** registers, etc.):
173             **      HEX mode.
174             **      XM=0.
175             **      See NOTE, below.
176             **
177             ** Normal exit conditions from handler if not handled (ST, RAM,
178             ** registers, etc.):
179             **      HEX mode.
180             **      XM=1.
181             **      See NOTE, below.
182             **
183             ** Available subroutine levels:
184             **      5
185             **
186             ** NOTE:
187             **      IMAGE parsing and execution are very involved.
188             **      Study the USING routine header and pIMbck, pIMcpi,
189             **      pIMXCH and pIMCHR poll documentation to learn
190             **      more about the process.  The USING routine header
191             **      describes the meaning and values of the IMAGE
```

```
192             **      tokens.
193             **
194             **      During parsing, the IMAGE string and BldIMG token
195             **      stream is kept in available memory, below AvMemEnd.
196             **      The BldIMG token stream is built backwads (toward
197             **      address 0) from the boundary of the IMAGE string.
198             **      At the time of the pIMXQT poll memory looks like this:
199             **
200             **                                                (Old AvMemEnd)
201             **      ------------------------------------------------
202             **          + | BldIMG tokens | IMAGE string   |
203             **      ---/------------------------------------------
204             **        /               ^               ^
205             **     cont'd below       |          'x'= last character parsed
206             **         xqt address in R0(9-5) points to execution token
207             **
208             **      ------------------------------------------------
209             **       | 5 nibs |3 nibs| 5 nibs | 5 nibs | 5 nibs | BldIMG...
210             **      ------------------------------------------------
211             **       ^     ^      ^        ^     ^         ^
212             **       |     |      |        |     |  length of IMAGE string
213             **  AvMemEnd   |  status bits  |   offset to 'x' above
214             **              |          offset to start of IMAGE string
215             **         5 zeros (stores offset
216             **         to xqt address when necessary)
217             **
218             **      IMAGE execution begins every time a new output field
219             **      is parsed, or when the end of the IMAGE string is
220             **      found.  By the time this poll occurs, all set-up
221             **      for execution has been performed (all pointers and
222             **      offsets have been stored away in AvMem).  R1(A)
223             **      contains the address of the IMAGE character which
224             **      caused execution to start (a specifier for a new
225             **      field, or a uIMend token).
226             **
227             **      What the poll handler does with the pIMXQT poll is
228             **      up to it.  The mainframe IMAGE execution routines
229             **      should serve for any type of output (DISP USING,
230             **      PRINT USING, OUTPUT USING, etc.), unless some
231             **      LEX file wants to output to some non-standard
232             **      device.  If so, it would pick up the IMAGE execution
233             **      at the pIMXQT poll and perform its own execution.
234             **
235             **      The most useful implementation of a pIMXQT poll
236             **      handler is for ENTER USING (found in the HPIL ROM).
237             **      The ENTER USING execution routines are vastly
238             **      different from the output routines, but use the
239             **      same IMAGE token streams.  Therefore, the ENTER
240             **      USING code intercepts the pIMXQT poll and performs
241             **      its own execution.
242             **
243             **      How the poll handler returns is also up to it.
244             **      In the case of ENTER USING, the poll handler jumps
245             **      directly back to entry point USGrst (restart parse),
246             **      without exiting through the poll code.  A poll
```

```
247             **      handler may exit through the poll code after
248             **      "handling" the poll, but it would want to adjust
249             **      pointers in R0 and possibly in RAM, also.
250             **
251             **      If exiting through USGrst:
252             **        -- RAM pointers, offsets and ST storage unchanged
253             **        -- R3(A)=Program Counter
254             **        -- other R registers unimportant
255             **      If exiting through poll code (XM=0):
256             **        -- RAM pointers, offsets and ST storage unchanged
257             **        -- R3(A)=Program Counter
258             **        -- R0(9-5)=xqt address, pointing to a uRESTP token
259             **
260             **
261             ** What registers/RAM may be used if handled?:
262             **      A,B,C,D,D0,D1,P,R1,R2
263             **      R0 (to adjust address of execution token)
264             **      R3 (to adjust Program Counter)
265             **
266             ** What registers/RAM may be used if not handled?:
267             **   If truly "not handled":
268             **      A,B,C,D[15-5],P,R2
269             **   If handled, but leaving XM=1:
270             **      A,B,C,D[15-5],P,R2
271             **
272             **
273             ** Special memory/pointer considerations (are pointers funny?):
274             **      None.  AvMem is available for writing to; this will
275             **      not disturb the IMAGE routines.
276             **
277             ** Envisioned application(s):
278             **      1) ENTER USING routines use the pIMXQT poll to override
279             **         the mainframe output code, instead executing code
280             **         which inputs variables using the BldIMG token stream.
281             **      2) A LEX file may "pre-parse" an IMAGE string (and
282             **         store it in an I/O buffer) for faster execution,
283             **         eliminating the need to parse the IMAGE string
284             **         every time the statement is executed.  It could
285             **         invoke the IMAGE parse routines and trap the
286             **         pIMXQT poll before execution starts.
287             **
288             ** History:
289             **
290             **    Date     Programmer            Modification
291             **  --------   ----------   ----------------------------------
292             **  12/08/82   MB           Documentation
293             **
294             ********************************************************************
295             ********************************************************************
```

```
296                 EJECT
297             ****************************************************************
298             ****************************************************************
299             **
300             ** Name:(S) pIMXCH  -  Unrecognized symbol in IMAGE execution
301             **
302             ** Category:   POLL
303             **
304             ** Type:       FPOLL
305             **
306             ** Purpose:
307             **      Allow LEX files to execute unrecognized IMAGE tokens.
308             **
309             ** Should poll be "Handled" (return with XM=0)?:
310             **      Yes.  If the poll is not handled by any LEX file,
311             **      the IMAGE routines issue an "Invalid USING" error.
312             **
313             ** Meaning of "Handling" Poll (what does code do if handled?):
314             **      The symbol was executed by a LEX file, generating
315             **      the appropriate output.
316             **
317             ** Entry conditions for handler (registers, ST, RAM, etc.):
318             **      Carry set.
319             **      B[A] = Poll number.
320             **      HEX mode.
321             **      P=0.
322             **      R0(A)=address of uIMXCH token which caused poll.
323             **      If within a numeric field:
324             **         R0(9-5)= counter for #zeroes in field
325             **         R0(S)= flag to identify last numeric symbol:
326             **                0= *
327             **                1= Z
328             **                5= D
329             **      R3(A)=Program Counter
330             **
331             ** Normal exit conditions from handler if handled (ST, RAM,
332             ** registers, etc.):
333             **      HEX mode.
334             **      XM=0.
335             **      R0(A)= address+2 of next token to execute
336             **         in BldIMG stream
337             **      S5=0
338             **      R3(A)=Program Counter
339             **      RAM storage above AvMemEnd untouched.
340             **
341             ** Normal exit conditions from handler if not handled (ST, RAM,
342             ** registers, etc.):
343             **      HEX mode.
344             **      XM=1.
345             **      R0,R3 and RAM storage above AvMemEnd untouched.
346             **
347             ** Available subroutine levels:
348             **      5
349             **
350             ** NOTE:
```

```
351             **      See NOTE under pIMXQT poll for RAM storage description.
352             **
353             **      The pIMXCh poll is issued only when a uIMXCH token
354             **      is encountered when executing the BldIMG tokens.
355             **      The uIMXCH token can only be placed by a poll handler
356             **      which previously handled a pIMCHR poll; their combined
357             **      purpose is to allow "strange" characters to be parsed
358             **      and executed in a IMAGE string.
359             **
360             **      The uIMXCH token in the BldIMG stream should be
361             **      accompanied by other tokens (or ASCII bytes) which
362             **      the poll handler will use for identification and
363             **      execution.
364             **
365             **      The pIMXCH poll is handled by the MATH ROM when
366             **      executing complex IMAGE fields.  The uIMXCH token is
367             **      inserted in the BldIMG stream in two places: 1) at
368             **      start of the complex field, so that the complex exp-
369             **      ression is evaluated, and a left parenthesis is out-
370             **      put, and 2) at the end of the field, to close out the
371             **      field and display a right parenthesis.  In the first
372             **      case a special token accompanies the uIMXCH token to
373             **      identify it to the MATH ROM as a complex field.  In
374             **      the second case, only an ASCII ")" accompanies the
375             **      uIMXCH token, which is all that is needed to signal
376             **      that the right parenthesis need be displayed.
377             **      For the two cases of complex fields using the
378             **      uIMXCH token, the partial tokenization looks
379             **      like this (it's built backwards towards address
380             **      zero):
381             **          case 1)
382             **                   uX  uC  u?  ...(existing BldIMG tokens)
383             **                  (3) (2) (1)
384             **
385             **          case 2)
386             **                   uX  =)  ...(existing BldIMG tokens)
387             **                  (5) (4)
388             **
389             **          where
390             **              uX = uIMXCH token
391             **              uC = uCPLXC token
392             **              u? = flag byte indicating "multiplied field"
393             **              =) = ASCII ")"
394             **
395             **      The code in the MATH ROM looks for the appropriate
396             **      byte values preceding the uIMXCH token to indicate
397             **      the appropriate action.
398             **
399             **      If a uIMXCH token has been inserted within a numeric
400             **      field, some extra steps have to be taken to insure
401             **      the float-check (for D symbols), and the skip-check
402             **      (for NaNs, Infs and overflows) are performed properly.
403             **
404             **      The float-check is performed to count the number of
405             **      positions that editing symbols or sign symbols must
```

```
406             **      float over leading zeroes (hence only performed for
407             **      the D fields).  The skip-check is performed to count
408             **      the number of positions to fill with spaces (for
409             **      NaN or INF) or *'s (for overflow).  If the new symbol
410             **      needs to be counted for either reason, you must
411             **      follow the uIMXCH token with a "D" or "S" or something
412             **      appropriate to cause the count to be incremented.
413             **      This extra "D" or "S" should be protected from the
414             **      execution routine; that is, the uIMXCH poll handler
415             **      should position the execution pointer (passed back
416             **      in R0(A)) past this extra character.  On the other
417             **      hand, to make the new symbol terminate either check,
418             **      insert an EndNum token as an extra character.
419             **      Both checks do not poll for uIMXCH; only the token
420             **      executor issues a poll.  Thus if the uIMXCH token
421             **      involves pointers which might look like any of the
422             **      symbols
423             **        D S X M . C Z P R  uMULT, uSTRPT or a byte>E5
424             **      you will have to protect it with uSTRPT (which
425             **      skips over 14 nibbles) or a uMULT (which skips
426             **      over 10 nibbles).
427             **
428             **      For instance, say the new character "I" is allowed
429             **      anywhere in an output field, having the same effect
430             **      as the "parent" symbols (the rest of the symbols
431             **      which define the type of field), except that the
432             **      character in that position is displayed in inverse
433             **      video.  For instance, "AAIA" is equivalent to "AAAA",
434             **      except that the third character is displayed in
435             **      inverse video.  Similarly, "DDID" is equivalent to
436             **      "DDDD", with an inverse video digit in the third
437             **      position.  Since "I" should be counted in the float-
438             **      check and skip-check (since it is allowed in a numeric
439             **      field), the (partial) token stream should look like
440             **      this (it's built backwards towards address 0), using
441             **      using "DDID" as an example:
442             **
443             **                      =D  =I  =D  uX  =D  =D
444             **                     (6) (5) (4) (3) (2) (1)
445             **
446             **      where
447             **          =D = ASCII "D"
448             **          =I = ASCII "I"
449             **          uX =uIMXCH token to cause pIMXCH poll.
450             **      Token (3) would be inserted by the poll handler for
451             **      a pIMCHR poll.  Then, during execution, the float-
452             **      check routine will count (4), and the pIMXCH poll
453             **      handler will execute (5) when the poll is issued
454             **      at (3).  When returning from the pIMXCH poll, the
455             **      execution pointer in R0(A) should be at (6).
456             **
457             **      Now say that the symbol "!<f,d>" causes a beep of
458             **      frequency f, duration d; the new symbol can be
459             **      inserted in any output field.  Then "DD!<800,.5>D"
460             **      would be tokenized as follows:
```

```
461             **
462             **              =D   uJ  p2  p1  uS =!   uX =D  =D
463             **             (9) (8) (7) (6) (5) (4) (3) (2) (1)
464             **       where
465             **           uJ =uJMPst (jumps 14 nibs on backward search)
466             **           p2 =5 nibble pointer to beep duration
467             **           p1 =5 nibble pointer to beep frequency
468             **           uS =uSTRPT (jumps 14 nibs in float-check)
469             **           uX =uIMXCH token, to cause pIMXCH poll
470             **           =D =ASCII "D"
471             **           =! =ASCII "!"
472             **       Then, during a float-check, (5) will cause a jump
473             **       over the pointers p1 and p2, to token (9); otherwise
474             **       these pointers might be interpreted as executing
475             **       tokens.  Token (8) is included for backward
476             **       searching during parse; it causes a jump over
477             **       pointers p1 and p2 for the same reason.  Token
478             **       (4) will be executed by the poll handler when the
479             **       pIMXCH poll is issued at (3).
480             **
481             ** What registers/RAM may be used if handled?:
482             **       A,B,C,D,D0,D1,P
483             **       R0 (to adjust pointer or counter)
484             **       R1 (to adjust counter)
485             **       R3 (to adjust Program Counter)
486             **
487             ** What registers/RAM may be used if not handled?:
488             **       A,B,C,D[15-5], D0, D1, P
489             **       R0, R3 untouched.
490             **       RAM storage above AvMemEnd untouched.
491             **       Expression stored in AvMem below AvMemEnd untouched.
492             **
493             ** Special memory/pointer considerations (are pointers funny?):
494             **       If the pIMXCH poll is issued while an output field is
495             **       pending (that is, the expression has already been
496             **       executed, but output not completed), the memory below
497             **       AvMemEnd contains the expression, and may not be
498             **       altered.
499             **
500             ** Envisioned application(s):
501             **       Complex IMAGE fields.
502             **       Some more are listed in NOTE, above.
503             **
504             ** History:
505             **
506             **    Date     Programmer             Modification
507             **   --------   ----------   ------------------------------------
508             **   12/08/82   MB           Documentation
509             **
510             ********************************************************************
511             ********************************************************************
```

```
512                     EJECT
513             ****************************************************************
514             ****************************************************************
515             **
516             ** Name:(S) pIMcpW  -  Working on complex image field
517             **
518             ** Category:   POLL
519             **
520             ** Type:       FPOLL
521             **
522             ** Purpose:
523             **      Alert MATH ROM to work on complex field.
524             **
525             ** Should poll be "Handled" (return with XM=0)?:
526             **      No.
527             **
528             ** Meaning of "Handling" Poll (what does code do if handled?):
529             **      Complex expression was evaluated, real or imaginary
530             **      part has been put on stack, ready for formatted out-
531             **      put.
532             **
533             ** Entry conditions for handler (registers, ST, RAM, etc.):
534             **      Carry set.
535             **      B[A] = Poll number.
536             **      HEX mode.
537             **      P=0.
538             **      R0(A)=address of numeric delimiter (in BldIMG token
539             **        stream) which caused the poll.
540             **      R3(A)=Program Counter
541             **
542             **
543             ** Normal exit conditions from handler if handled (ST, RAM,
544             ** registers, etc.):
545             **   This poll can only be handled by the MATH ROM.  It
546             **   cannot exit through the poll routines with XM=0;
547             **   it can only exit by jumping to USnmO5.
548             **      HEX mode.
549             **      A(W)=numeric expression (either the real or imaginary
550             **        part, as appropriate)
551             **      D1 points to AvMemEnd-16.
552             **      R registers untouched.
553             **
554             ** Normal exit conditions from handler if not handled (ST, RAM,
555             ** registers, etc.):
556             **      HEX mode.
557             **      XM=1.
558             **      R registers untouched.
559             **
560             ** Available subroutine levels:
561             **      7 (junk the two poll levels, and jump to USnmO5)
562             **
563             ** NOTE:
564             **      This poll can only be handled by the MATH ROM, as
565             **      part of complex image field execution.
566             **
```

```
567             ** What registers/RAM may be used if handled?:
568             **      A,B,C,D,D0,D1,P,R0(15-5),R1,R2,R3(9-5),R4
569             **      R0(A) should not be used
570             **      R3(A) should not be used
571             **
572             ** What registers/RAM may be used if not handled?:
573             **      A,B,C,D[15-5],D0,D1,P,R1,R2,R4
574             **
575             ** Special memory/pointer considerations (are pointers funny?):
576             **      At the time of the poll, AvMem is not used to store
577             **      anything.  If the poll is handled properly, the
578             **      expression for output resides at AvMemEnd-16.
579             **
580             ** Envisioned application(s):
581             **      MATH ROM complex field output.  Only.
582             **
583             ** History:
584             **
585             **    Date     Programmer             Modification
586             **  --------   ----------   ----------------------------------
587             **  01/01/83   MB           Implemented, documented.
588             **
589             ********************************************************************
590             ********************************************************************
```

```
591                 EJECT
592             ********************************************************************
593             ********************************************************************
594             **
595             ** Name:    IMGxqt  -  Execute IMAGE string
596             ** Name:    IMGxq1  -  Execute IMAGE string
597             **
598             ** Category:   STEXEC
599             **
600             ** Purpose:
601             **      Main entry point for executing IMAGE tokens;
602             **      tokenization is performed by routines in MB&IMG.
603             **
604             ** Entry:
605             **      IMGxqt: P=0
606             **      IMGxq1: D1=current position in BldIMG stream
607             **              R0(A)=same value as D1
608             **              R0(9-5)=address to begin executing fields
609             **              R1(A)=address of last parsed IMAGE symbol
610             **              R1(9-5)=length of IMAGE string (in nibbles)
611             **              R3(A)=Program Counter
612             **              R3(9-5)=Address of start of IMAGE string.
613             **
614             ** Exit:
615             **      Normally exits through USGrst routine.
616             **
617             ** Calls:      IMAGE execution routines call EXPEXC, which can
618             **             call anything.
619             **
620             ** Uses.......
621             **  Inclusive: IMAGE execution routines call EXPEXC, which can
622             **             use anything.
623             **
624             ** Stk lvls:   IMAGE execution routines call EXPEXC, which can
625             **             use all 7 stack levels.
626             **
627             ** NOTE:
628             **      IMGxqt is the standard entry point for executing pend-
629             **      ing output fields in the IMAGE string.  Pending output
630             **      fields are executed whenever a new output field is
631             **      encountered; a uRESTP (restart parse) token is inserted
632             **      into the BldIMG stream just before falling into IMGxq1.
633             **
634             **      IMGxq1 is the entry point for executing IMAGE fields.
635             **      The last token to be executed should be either a uRESTP
636             **      (restart parse) token (which halts execution and begins
637             **      parsing where it left off before), or a uIMend (end-of-
638             **      image) token (which tests the output list to see if it
639             **      is exhausted).  This entry point is used by the parse
640             **      routines encounter the end of the IMAGE string, and by
641             **      poll handlers for pIMCHR which have detected the start
642             **      of a new output field.
643             **
644             ** Detail:
645             **  When execution starts, a save area is set up at the
```

```
646             **  bottom (low address) of the stack, below the expanded
647             **  BldIMG string:
648             **
649             **           5 nibs 3 nibs 5 nibs 5 nibs  5 nibs
650             **    ---------------------------------------------
651             **          |      |    |      |      |      | BldIMG...
652             **  / ---------------------------------------------
653             ** used for  \   ^    ^      ^     ^       ^
654             ** expr only  |   |    |      |     |  length of IMAGE string
655             **     AvMemEnd |    |      |    offset to DO pointer
656             ** offset to xqt addr  |  offset to start of IMAGE string
657             **                  status bits
658             **
659             **
660             **   Available-memory-end is set to the storage location of the
661             **   offset to the execution address, and remains there through
662             **   Expression Execute.
663             **
664             **   Memory below AvMemEnd is used only to store an expression
665             **   (put there by a call to EXPEXC).  After return from
666             **   EXPEXC, the offset to the execution address is replaced
667             **   by the offset to the next character to output (or in the
668             **   case of H,K,B or ^, by the offset to the beginning of the
669             **   stack item).
670             **
671             **    Just before calling EXPR:              5 nibs
672             **      ---------------------------------------------
673             **                                        |      |
674             **      ---------------------------------------------
675             **                              AvMemEnd --+  /
676             **                             offset to xqt addr
677             **
678             **    After returning from EXPR:
679             **      ---------------------------------------------
680             **        |                               |      |
681             **      ---------------------------------------------
682             **                        /       AvMemEnd --+ /
683             **    expr put onto stack       offset into expr
684             **
685             **
686             ** History:
687             **
688             **    Date     Programmer             Modification
689             **  --------   ----------   ---------------------------------
690             **  12/08/82   MB           Documentation
691             **
692             ****************************************************************
693             ****************************************************************
694             *
695 1BAA4      =IMGxqt                        IMAGE execute.
696 1BAA4 311F         LC(2)  uRESTP          Change Delim to "Restart Parse".
697 1BAA8 14D          DAT1=C B
698 1BAAB 110  =IMGxq1 A=R0                   A= end of BldIMG.
699             *
700 1BAAE 119         C=R1                   C(A)=IMAGE ptr, length to C(9-5).
```

```
701 1BAB1 29          P=     9            10 nibs to BldIMG.
702 1BAB3 7665        GOSUB  DT1C.A       Write D0 offset to buffer.
703               *
704 1BAB7 118         C=R3
705 1BABA 71C5        GOSUB  CSRW9j       Addr start of IMAGE to C(A).
706 1BABE 7655        GOSUB  DT1C-A       Write offset to buffer.
707               *
708 1BAC2 740F        GOSUB  =CLOST+      Clear 0 nib of st, clear sSpecl.
709 1BAC6 7C3F        GOSUB  =CSLW9j      St bits to C(7-5).
710 1BACA 27          P=     7            8 nibs to BldIMG.
711 1BACC 7A9F        GOSUB  =BldIM+      St bits, 5 zeros to BldIMG.
712               *
713 1BAD0 8E00        GOSUBL =AVE=D1      Set AvMemEnd=D1.
          00
714               *
715 1BAD6 7D8E        GOSUB  =FPOLLj      "Start IMAGE execution" poll.
716 1BADA 00          CON(2) =pIMXQT
717               *                       Handled or not, I'm gonna execute.
718               *                       If handled, assume R0(9-5)= addr
719               *                         of uRESTP token.
720 1BADC 7C95 =IMxqQ5 GOSUB  R0u2CA       R0(9-5)=xqt address to C(A).
721 1BAE0 135         D1=C
722 1BAE3 171         D1=D1+ 2            Undo next D1-2.
723               *
```

```
                EJECT
*****************************************************************
*****************************************************************
**
** Name:    IMxq12  -  Main execution loop for IMAGE tokens
**
** Category:   EXCUTL
**
** Purpose:
**      Main execution loop for IMAGE tokens.  Jumps to
**      appropriate execution routine according to token value.
**
** Entry:
**      D1=address+2 of next token to execute.
**
** Exit:
**      Exits through appropriate execution routine.
**
** Calls:      May call EXPEXC, which can use anything.
**
** Uses.......
**  Inclusive: May call EXPEXC, which can use anything.
**
** Stk lvls:   May call EXPEXC, which can use anything.
**
** Detail:
**      IMxq12 is followed by a table of token values and
**      relative offsets to execution routines.  IMxq12 calls
**      BYTscA, which scans the table for the appropriate jump.
**
** History:
**
**    Date     Programmer              Modification
**  --------   ----------   ------------------------------------
**  12/08/82   MB           Documentation
**
*****************************************************************
*****************************************************************
*
*****************************************************************
*****************************************************************
**
** Name:(S) IMxq27  -  Return to IMAGE token executor
**
** Category:   EXCUTL
**
** Purpose:
**      Return to IMxq12 (main IMAGE token execution routine)
**      after restoring D1 (token pointer).
**
** Entry:
**      C(A)=address+2 of next IMAGE token to execute.
**      S5=0
**      S6=0
**
```

```
779             ** Exit:
780             **      May jump to any execution routines.
781             **
782             ** Calls:      May jump to any execution routines.
783             **
784             ** Uses.......
785             **  Inclusive: May jump to any execution routines.
786             **
787             ** Stk lvls:   May jump to any execution routines.
788             **
789             ** NOTE:
790             **      Some IMAGE poll handlers will use this entry point
791             **      after handling a poll.  Since the FPOLL routine does
792             **      not preserve D1, this allows a poll handler to jump
793             **      to the IMAGE token executor with D1 pointing to the
794             **      appropriate token.
795             **
796             ** History:
797             **
798             **    Date     Programmer             Modification
799             **  --------   ----------   ----------------------------------
800             **  12/08/82   MB           Documentation
801             **
802             ********************************************************************
803             ********************************************************************
804             *
805 1BAE6 845    IMxq10 ST=0   sRturn         Clear Return flag for USGstr.
806 1BAE9 866           ?ST=0  sSpecl         Outputting NaN or Inf?
807 1BAEC 90            GOYES  IMxq12         No.
808             *
809 1BAEE 6AF6          GOTO   CHKSK3         Yes. Check for skips.
810             *
811 1BAF2 171    IMxq11 D1=D1+ 2
812 1BAF5 715F   IMxq12 GOSUB  =BYTscA        Execute next token.
813             *
814 1BAF9 44            CON(2) \D\            Output digit (blank filled).
815 1BAFB 314           REL(3) USG"D"
816             *
817 1BAFE A5            CON(2) \Z\            Output digit (zero filled).
818 1BB00 C24           REL(3) USG"Z"
819             *
820 1BB03 14            CON(2) \A\            Output ASCII char.
821 1BB05 321           REL(3) USG"A"
822             *
823 1BB08 1D            CON(2) uMULT          Multiplier.
824 1BB0A 3A0           REL(3) USGmlt
825             *
826 1BB0D 2D            CON(2) uLOOPB         Loop on byte.
827 1BB0F 5A0           REL(3) USGlpb
828             *
829 1BB12 85            CON(2) \X\            Output ASCII blank.
830 1BB14 7D0           REL(3) USGch4
831             *
832 1BB17 E2            CON(2) \.\            Output "." (radix).
833 1BB19 0D0           REL(3) USGch5
```

```
834                *
835 1BB1C A2           CON(2) \*\          Output digit (* filled).
836 1BB1E 514          REL(3) USG"*"
837                *
838 1BB21 35           CON(2) \S\          Output sign (+ or -).
839 1BB23 664          REL(3) USG"S"
840                *
841 1BB26 D4           CON(2) \M\          Output sign (blank or -).
842 1BB28 364          REL(3) USG"M"
843                *
844 1BB2B 34           CON(2) \C\          Output "," (separator).
845 1BB2D C34          REL(3) USG"C"
846                *
847 1BB30 05           CON(2) \P\          Output "." (separator).
848 1BB32 934          REL(3) USG"P"
849                *
850 1BB35 7F           CON(2) uALit    Delimiter: string literal.
851 1BB37 011          REL(3) USGlit
852                *
853 1BB3A 84           CON(2) \H\          Output compact form (European).
854 1BB3C 451          REL(3) USG"H"
855                *
856 1BB3F B4           CON(2) \K\          Output compact form.
857 1BB41 C41          REL(3) USG"K"
858                *
859 1BB44 54           CON(2) \E\          Output "E" (exponent).
860 1BB46 A90          REL(3) USG"E"
861                *
862 1BB49 0D           CON(2) uSTRPT       Pointer to imbedded literal.
863 1BB4B 470          REL(3) USGstr
864                *
865 1BB4E 3D           CON(2) uLOOPS       Loop on string.
866 1BB50 660          REL(3) USGlps
867                *
868 1BB53 FE           CON(2) uLOOPP       Loop on parentheses.
869 1BB55 360          REL(3) USGlpp
870                *
871 1BB58 1F           CON(2) uRESTP       Restart parse.
872 1BB5A 901          REL(3) USGrst
873                *
874 1BB5D 4F           CON(2) uDELIM       Unfilled delimiter.
875 1BB5F DF0          REL(3) USGdlm
876                *
877 1BB62 0F           CON(2) uIMend       IMAGE end.
878 1BB64 C85          REL(3) USGend
879                *
880 1BB67 24           CON(2) \B\          Output byte form.
881 1BB69 274          REL(3) USG"B"
882                *
883 1BB6C F2           CON(2) \/\          Output EOL sequence.
884 1BB6E 746          REL(3) USGch/
885                *
886 1BB71 25           CON(2) \R\          Output "," (radix).
887 1BB73 470          REL(3) USGch6
888                *
```

```
 889 1BB76 04          CON(2) \@\          Output Form Feed.
 890 1BB78 D60         REL(3) USGch7
 891             *
 892 1BB7B 4D          CON(2) uIMXCH       Poll for unrecognized xqt char.
 893 1BB7D 210         REL(3) IMXPOL
 894             *
 895 1BB80 00          CON(2) 0            Others: check for numeric delims.
 896 1BB82 316F        LC(2)  uHKB^        See table below for tokens--+
 897 1BB86 9EE         ?A>=C  B            Numeric delimiter?          |
 898 1BB89 02          GOYES  ToUSGn       Yes. To USGnum.             |
 899 1BB8B 6A5F  IMxq19 GOTO  IMxq10       Execute next token.         |
 900             *                                                     |
 901             *  +--------------------------------------------------+
 902             *  |   F6  uHKB^     Delimiter: H,K,B or ^ takes numeric...  |
 903             *  |  (F7  uALit  Already trapped in table above.)           |
 904             *  |   F8  uNUMNn    Delimiter: Numeric, no float, no sign.  |
 905             *  |   F9  uNUMNs    Delimiter: Numeric, no float, w/sign.   |
 906             *  |   FA  uNUMFn    Delimiter: Numeric, float, no sign.     |
 907             *  |   FB  uNUMFs    Delimiter: Numeric, float, w/sign.      |
 908             *  |   FC  uNUMEn    Delimiter: Numeric, w/exponent, no sign.|
 909             *  |   FD  uNUMEs    Delimiter: Numeric, w/exponent, w/sign. |
 910             *  |   FE  <<nothing>>   Checks anyway...                    |
 911             *  |   FF  <<nothing>>   Checks anyway...                    |
 912             *  +--------------------------------------------------+
 913             ************************************************************
 914             *
 915             *
 916 1BB8F       IMXPOL                      Unrecognized xqt char poll.
 917 1BB8F 700E          GOSUB  =FPOLLx      Pass D1=xqt addr in R0(A), poll.
 918 1BB93 00          CON(2) =pIMXCH
 919 1BB95 7DED =IMXRTN GOSUB  =Polrtn       Check for poll handled.
 920             *          +-----------------------------------------+
 921             *          | If poll handled, handler must return:   |
 922             *          |    R0(A)=xqt address (passed to you in R0(A)) |
 923             *          +-----------------------------------------+
 924             *
 925 1BB99 118   IMxq25 C=R0                 C= addr IMAGE execution.
 926 1BB9C 135  =IMxq27 D1=C                 D1= addr IMAGE execution.
 927 1BB9F 865         ?ST=0  sRtrn         Return from displaying char?
 928 1BBA2 9E          GOYES  IMxq19        Yes.
 929             *
 930 1BBA4 845         ST=0   sRtrn         Clear "Return" flag.
 931 1BBA7 03          RTNCC
 932             *
 933 1BBA9 6E41  ToUSGn GOTO   USGnum
 934             *
```

```
935                     EJECT
936               ************************************************
937               *
938 1BBAD 76C5  USGmlt GOSUB  DCRMNT          Decrement multiplier counter.
939 1BBB1 59D           GONC   IMxq19         (BET) Next xqt symbol.
940               *
941               ************************************************
942               *
943 1BBB4 24    USGlpb P=     4               Loop on byte. 4 nib jump.
944               *
945 1BBB6 0D    USGlps P=P-1                  Loop on string ptr. 16 nib jump.
946 1BBB8 7F85  USGlpp GOSUB  USloop          D1 to loop multiplier, decrement.
947 1BBBC 5EC           GONC   IMxq19         (BET) Next xqt symbol.
948               *
```

```
                EJECT
*****************************************************************
*****************************************************************
**
** Name:(S) USst03  -  Output characters from address in C
** Name:(S) USst05  -  Output characters from address in D1
**
** Category:   EXCUTL
**
** Purpose:
**      To output a character during USING execution; character
**      display observes WIDTH.
**
** Entry:
**      USst03: D1=address of current token being executed
**              C=address of characters to be output
**      USst05: A=address of current token being executed
**              D1=address of characters to be output
**         P=0
**         B(A)=#characters to output
**         CKINFO must have been called previously to set up
**           the output information (see CKINFO)
**         S5=0 to exit to IMxq12, S5=1 to return.
**
** Exit:
**      P      =  0
**      If S5=0, exits to IMxq12
**      If S5=1, does a "return", carry clear.
**
** Calls:      SENDWD
**
** Uses.......
**  Exclusive: A,B,C,R0,S4
**  Inclusive: A,B,C,D,R0,R1,R2,P,S4,D1
**
** Stk lvls:   5
**
** NOTE:
**      If you want to display only one character, call USGch+
**
** Detail:
**      Before call to SENDWD, sets S4=0 to inhibit EOL before
**      item is displayed.
**
**     =USst03 AD1EX
**              D1=C
**     =USst05 R0=A
**              A=B    A
**              ST=0   4
**              GOSBVL =SENDWD
**              C=R0
**              D1=C
**              ?ST=0  5
**              GOYES  IMxq12
**              ST=0   5
```

```
1004             **             RTNCC
1005             **
1006             ** History:
1007             **
1008             **    Date      Programmer            Modification
1009             **  --------    ----------   ----------------------------------
1010             **  12/08/82    MB           Documentation
1011             **
1012             ********************************************************************
1013             ********************************************************************
1014             *
1015 1BBBF 110   USGstr A=R0                    (Preserve R0(9-5)=zero counter)
1016 1BBC2 7A84         GOSUB  RCVOFS          Recover offset to literal.
1017 1BBC6 1C9          D1=D1- 10              D1 points to literal length.
1018 1BBC9 143          A=DAT1 A               Length of literal.
1019 1BBCC D8           B=A    A               Length to B.
1020 1BBCE 133  =USst03 AD1EX                  D1 (xqt addr) to A.
1021 1BBD1 135          D1=C                   Address of literal to D1.
1022 1BBD4 100  =USst05 R0=A                   Save D1 xqt addr in R0.
1023 1BBD7 D4           A=B    A               Length of literal to A.
1024 1BBD9 7240         GOSUB  SENDWj          Send, in width-sized chunks.
1025 1BBDD 4BB          GOC    IMxq25          (BET) To next execution symbol.
1026             *
```

```
1027                    EJECT
1028                ****************************************************************
1029                ****************************************************************
1030                **
1031                ** Name:(S) USGch+  -  Display character during USING execution
1032                ** Name:(S) USGch-  -  Display character during USING execution
1033                **
1034                ** Category:   EXCUTL
1035                **
1036                ** Purpose:
1037                **      To display one character during USING execution.
1038                **
1039                ** Entry:
1040                **      USGch-: RSTK address contains table of ASCII characters
1041                **              P=pointer into ASCII table
1042                **      USGch+: P=0
1043                **              C(A)=address of ASCII character
1044                **         D1=address of current IMAGE token being executed.
1045                **
1046                ** Exit:
1047                **      See USst03
1048                **
1049                ** Calls:      USst03
1050                **
1051                ** Uses.......
1052                **  Exclusive: A(W),B(A),C(A),P
1053                **  Inclusive: A,B,C,D,R0(A),R1,R2,P,D1
1054                **
1055                ** Stk lvls:   5
1056                **
1057                ** NOTE:
1058                **      For USGch- entry, the ASCII table must have a 00 byte
1059                **      as the first entry.  A value of P=0 would point to the
1060                **      first byte past this 00 byte.
1061                **
1062                ** Detail:
1063                **     =USGch- C=RSTK            Address of ASCII table.
1064                **             C+P+1             Pointer into table.
1065                **             C+P+1
1066                **             P=      0
1067                **     =USGch+ B=0     A         B(A)=1=#characters
1068                **             B=B+1   A            to display.
1069                **             A=R0              Preserve R0(9-5).
1070                **             GOTO    USst03
1071                **
1072                ** History:
1073                **
1074                **    Date     Programmer             Modification
1075                **  --------   ----------   ----------------------------------
1076                **  12/08/82   MB           Documentation
1077                **
1078                ****************************************************************
1079                ****************************************************************
1080 1BBE0 852    USG"E" ST=1   sNUME          Output exponent digits.
1081 1BBE3 0C            P=P+1                 Reaches USGch0 with P=8.
```

```
1082            ************************************************
1083 1BBE5 0C    USGch7 P=P+1                  (P=7) Form Feed ("@")
1084 1BBE7 0C    USGch6 P=P+1                  (P=6) ","
1085 1BBE9 0C    USGch5 P=P+1                  (P=5) "."
1086 1BBEB 0C    USGch4 P=P+1                  (P=4) " "
1087 1BBED 0C    USGch3 P=P+1                  (P=3) "-"
1088 1BBEF 0C    USGch2 P=P+1                  (P=2) "+"
1089 1BBF1 0C    USGch1 P=P+1                  (P=1) "0"
1090 1BBF3 7410  USGch0 GOSUB  USGch-          (P=0) "*"
1091            *
1092            *!!!! THIS NEXT BYTE IS FREE TO BE USED FOR ROM CHECKSUM !!!!
1093 1BBF7 00           NIBHEX 00             Filler for C+P+1, C+P+1.
1094 1BBF9 A2           NIBASC \*\            USGch0
1095 1BBFB 03           NIBASC \0\            USGch1
1096 1BBFD B2           NIBASC \+\            USGch2
1097 1BBFF D2           NIBASC \-\            USGch3
1098 1BC01 02           NIBASC \ \            USGch4
1099 1BC03 E2           NIBASC \.\            USGch5
1100 1BC05 C2           NIBASC \,\            USGch6
1101 1BC07 C0           NIBHEX C0             USGch7   Form Feed.
1102 1BC09 54           NIBASC \E\            USGch8
1103            *
1104            *
1105 1BC0B 07   =USGch- C=RSTK                 Add offset to desired
1106 1BC0D 809          C+P+1                    character.
1107 1BC10 809          C+P+1
1108 1BC13 20    USGch* P=     0               Reset pointer.
1109            *
1110 1BC15 D1   =USGch+ B=0    A
1111 1BC17 E5           B=B+1  A              "One char out".
1112 1BC19 110          A=R0                  (Preserve R0(9-5)=zero counter)
1113 1BC1C 51B          GONC   USst03         (BET)  Send out char.
1114            *
1115            ***************
1116 1BC1F 854   SENDWj ST=1   InhEOL
1117 1BC22 8C00         GOLONG =SENDWD
         00
1118            *
```

```
1119                    EJECT
1120              ************************************************
1121              *
1122 1BC28          USG"A"                      "A" symbol: send ASCII char.
1123 1BC28 7944             GOSUB  StAVE+       A=curr xqt addr, D1= AvMemEnd.
1124 1BC2C 147              C=DAT1 A            Read #nibs in string expr.
1125 1BC2F CE               C=C-1  A            Decrement for 1 char sent out.
1126 1BC31 4A0              GOC    USGA01       CRY= entire string sent.
1127 1BC34 CE               C=C-1  A            -2 nibs for each char.
1128 1BC36 145              DAT1=C A            Re-write counter.
1129 1BC39 1C1              D1=D1- 2            D1 to start of string.
1130 1BC3C 133    USGA01    AD1EX               D1= current xqt address.
1131 1BC3F 4BA              GOC    USGch4       From USGA1+. Send out space.
1132 1BC42 EE               C=A-C  A            C= address of char to send out.
1133 1BC44 50D              GONC   USGch+       (BET) Send out one char.
1134              *
1135              ****************************************
1136              *
1137 1BC47          USGlit                      "String delimiter" found.
1138 1BC47 843              ST=0   sZeros       GetExp allow only string.
1139 1BC4A 7834             GOSUB  GetEXP       Put string expression on stack,
1140              *                               store xqt address at AvMemEnd.
1141 1BC4E 8F00             GOSBVL =REVPOP      Reverse for output; A(A)= length.
          000
1142 1BC55 7DC3             GOSUB  NwOFF+       Store string length at AvMemEnd;
1143              *                               set C= xqt address.
1144 1BC59 135              D1=C                D1 points to xqt address.
1145 1BC5C 1C7    USGdlm    D1=D1- 8            Skip over delimiter zeroes.
1146 1BC5F 668E             GOTO   IMxq10       Next token.
1147              *
```

```
1148                    EJECT
1149                ****************************************************************
1150                ****************************************************************
1151                **
1152                ** Name:(S) USGrst  -  Suspend USING execution, restart parse
1153                **
1154                ** Category:   EXCUTL
1155                **
1156                ** Purpose:
1157                **      Halt IMAGE execution and restart parsing of IMAGE
1158                **      fields.
1159                **
1160                ** Entry:
1161                **      P      =  0
1162                **      R3(A)=Program Counter
1163                **      RAM storage at AvMemEnd is as shown in IMGxqt header.
1164                **
1165                ** Exit:
1166                **      To Nxtfl3 (parse next field).
1167                **
1168                ** Calls:      GETSTA, C+A2D1, R2=D1+, CA2D1., IMDO--, Nxtfl3
1169                **
1170                ** Uses.......
1171                **  Exclusive: A(A),C,D1
1172                **  Inclusive: IMAGE parse routines at Nxtfl3 can use anything
1173                **
1174                ** Stk lvls:   2 (before exit to Nxtfl3, which can use all 7)
1175                **
1176                ** NOTE:
1177                **      Most pIMXQT poll handlers will return to USGrst, after
1178                **      they have taken care of their execution.
1179                **
1180                ** Algorithm:
1181                **      Restore status bits from RAM.
1182                **      Restore address of start of IMAGE string to R3(9-5)
1183                **      Restore length of IMAGE string to R0(9-5)
1184                **      Restore address of next parse symbol to R0(A).
1185                **
1186                ** History:
1187                **
1188                **    Date     Programmer            Modification
1189                **  --------   ----------   ----------------------------------
1190                **  12/08/82   MB           Documentation
1191                **
1192                ****************************************************************
1193                ****************************************************************
1194                *
1195 1BC63          =USGrst                          Restart parse.
1196 1BC63 7CF3           GOSUB  GETSTA               Get status bits from storage.
1197 1BC67 172            D1=D1+ 3                    D1->offset to start IMAGE string.
1198                *
1199 1BC6A 75E3           GOSUB  C+A2D1               C=addr of start of IMAGE string.
1200 1BC6E DA             A=C    A                    Save addr in A.
1201 1BC70 11B            C=R3                        C=D0 pointer (PC).
1202 1BC73 8F00           GOSBVL =R3=D1+              Store both R3 (just like entry).
```

```
            000
1203 1BC7A 15F9         C=DAT1 10           C(9-5)=length IMAGE string,
1204              *                           C(A)= offset to restart D0.
1205 1BC7E 179          D1=D1+ 10           D1 past offset storage.
1206 1BC81 74D3         GOSUB  CA2D1.       C(A)= D0 pointer to IMAGE string.
1207 1BC85 7B9D         GOSUB  =IMD0--      Save both in R1.
1208 1BC89 6C49         GOTO   =Nxtf13      Restart parse.
1209              *
```

```
1210                    EJECT
1211              ******************************************************
1212              *
1213 1BC8D 853    USG"K" ST=1   sKnotH
1214 1BC90        USG"H"                          (sKnotH =0)
1215 1BC90 7D63           GOSUB  POPTS+           Test stack item.
1216 1BC94 501            GONC   UShk05           NC= numeric.
1217 1BC97 B04            A=A+1  P                No. Test for string.
1218 1BC9A 96C            ?A#0   B                String?
1219 1BC9D 80             GOYES  UShk09           No. Complex.
1220 1BC9F 853            ST=1   sKnotH           Omits test for "." (European...)
1221 1BCA2 580            GONC   UShk15           (BET)
1222              *
1223 1BCA5        UShk05
1224              *
1225 1BCA5 8E00   UShk09 GOSUBL =STR$SB           Convert number to display format.
           00
1226              *
1227 1BCAB AF0    UShk15 A=0    W                 (For SRB below.)
1228 1BCAE 8F00           GOSBVL =REVPOP          Reverse for output; A(A)=length.
           000
1229 1BCB5 873            ?ST=1  sKnotH           "K" specifier?
1230 1BCB8 D2             GOYES  UShk31           Yes.
1231 1BCBA 130            D0=A                    No. D0=#nibs in item (=cntr).
1232 1BCBD 137            CD1EX                   Addr start of item to C.
1233 1BCC0 C2             C=A+C  A                Compute end addr of item.
1234 1BCC2 135            D1=C                    D1= end addr.
1235 1BCC5 31E2           LCASC  \.\              Looking for "." radix.
1236 1BCC9 D7             D=C    A                Save "." in D.
1237              *
1238 1BCCB 181    UShk21 D0=D0- 2                 Count chars in string.
1239 1BCCE 461            GOC    UShk31           End of string.
1240 1BCD1 1C1            D1=D1- 2                To next char.
1241 1BCD4 14F            C=DAT1 B                Read char.
1242 1BCD7 967            ?C#D   B                "."?
1243 1BCDA 1F             GOYES  UShk21           No. To next char.
1244 1BCDC 30C            LC(1)  \,\              Yes. Replace with European ",".
1245 1BCDF 14D            DAT1=C B
1246 1BCE2 58E            GONC   UShk21           (BET)
1247              *
1248 1BCE5 81C    UShk31 ASRB                    A= #chars in item (#bytes).
1249 1BCE8 733F           GOSUB  SENDWj           Send to output device.
1250 1BCEC 7D33   UShk35 GOSUB  NWOFFS           Set C=addr of IMAGE execution.
1251 1BCF0 6BAE           GOTO   IMxq27           Next IMAGE symbol.
1252              *
```

```
1253                EJECT
1254            ****************************************************************
1255            ****************************************************************
1256            **
1257            ** Name:    USGnum  -  Evaluate and execute numeric IMAGE field
1258            ** Name:(S) USnn05  -  Execute numeric IMAGE field
1259            **
1260            ** Category:   EXCUTL
1261            **
1262            ** Purpose:
1263            **      To evaluate (through EXPEXC) and execute numeric IMAGE
1264            **      field.
1265            **
1266            ** Entry:
1267            **   RAM locations as specified in IMGxqt header.
1268            **   USGnum:
1269            **      P=0
1270            **      D1=address of current token in BldIMG stream
1271            **      A(B)=delimiter token which defined numeric field
1272            **   USnn05:
1273            **      P=0
1274            **      A(W)=numeric expression (real or imaginary part)
1275            **      D1 points to AvMemEnd-16, which also contains a copy
1276            **        of the expression in A.
1277            **
1278            ** Exit:
1279            **      Exits to IMxq12.
1280            **
1281            ** Calls:      SET-ST, FPOLL (pIMcpw), GetEXP, C+A2D1, DECP=C,
1282            **             RND-12, ExpEXP, CHKFLT
1283            **
1284            ** Uses.......
1285            **  Inclusive: GetEXP calls EXPEXC, which may use anything
1286            **
1287            ** Stk lvls:   GetEXP calls EXPEXC, which may use all 7
1288            **
1289            ** NOTE:
1290            **      USGnum is the routine which formats all numeric fields.
1291            **      The value of the delimiting token determines the status
1292            **      bit settings, which in turn define the type of format-
1293            **      ting (sign field, exponent field, etc.).
1294            **
1295            **      USnn05 is a return point for the pIMcpw poll ("complex
1296            **      field working").
1297            **
1298            ** Algorithm:
1299            **      Set status bits as specified by numeric delimiter.
1300            **      Fetch expression, store at AvMemEnd-16.
1301            **      Copy expression to B.
1302            **      Read #digits in field, store in D.
1303            **      Read #digits before radix, store in C.
1304            **      Allow 1 digit position for sign, if sign not specified.
1305            **      Expand exponent to 5 digit form.
1306            **      Calculate #zeroes before first nonzero digit.
1307            **      Calculate position to round; round expression.
```

```
1308            **      If exponent changed in rounding, decrement #zeroes.
1309            **      If insufficient digits, "IMAGE Ovfl" warning/error.
1310            **      Store #zeroes in R1.
1311            **      Store rounded expression back in AvMemEnd-16.
1312            **      If floating field (D's), go to CHKFLT
1313            **        else go to IMxq12.
1314            **
1315            ** History:
1316            **
1317            **    Date     Programmer            Modification
1318            **  --------   ----------   ---------------------------------
1319            **  12/08/82   MB           Documentation
1320            **
1321            ****************************************************************
1322            ****************************************************************
1323            *
1324 1BCF4 6BD4  NANINF GOTO   CHKSKP         Check for # skip posns.
1325            *
1326 1BCF8 73DC  USGnum GOSUB  =SET-ST        Set status bits according
1327            *                              to token value: s"S,M" ,
1328            *                              sFLOAT, sNUME .
1329 1BCFC 867          ?ST=0  sCplxW         Working on complex field?
1330 1BCFF C0           GOYES  USnm03         No. Get expression.
1331 1BD01 7E8C         GOSUB  =FPOLLx        "Work on complex number" poll.
1332            *                             Pass D1 to handler in R0(A).
1333 1BD05 00           CON(2) =pIMcpw
1334 1BD07 7B7C         GOSUB  =Polrtn        Check for poll handled,return.
1335            *--!!! NOTE: MATH ROM returns to USnm05 when poll handled!!!
1336            *          +-----------------------------------------------+
1337            *          |  If poll handled, handler must return:        |
1338            *          |    A(W)=numeric expression                    |
1339            *          |    D1 points to expression, which  resides    |
1340            *          |       at AvMemEnd-16  (i.e., D1=AvMemEnd-16)  |
1341            *          |       (Note: nothing important can reside     |
1342            *          |       below this location, since a call to    |
1343            *          |       STR$SB will certainly overwrite it.)    |
1344            *          +-----------------------------------------------+
1345            *
1346 1BD0B 856   USnm03 ST=1   sSpecl         "Accept numeric expression."
1347 1BD0E 7473         GOSUB  GetEXP         Get expression.
1348 1BD12 863  =USnm05 ?ST=0  sZeros         H,K,B,^ field?  (sZeros=sKnotH)
1349 1BD15 7D           GOYES  UShk35         Yes. No formatting.
1350            *                             No. Numeric.
1351 1BD17 AF8          B=A    W              Expression to B.
1352 1BD1A 17F          D1=D1+ 16             Set D1= AvMemEnd.
1353 1BD1D 7233         GOSUB  C+A2D1         C= current xqt address in BldIMG.
1354 1BD21 135          D1=C                  D1 points to numeric symbol.
1355 1BD24 1C7          D1=D1- 8              D1 points to #digits in field.
1356 1BD27 D4           A=B    A              Exponent to A.
1357 1BD29 B24          A=A+1  XS
1358 1BD2C 47C          GOC    NANINF         NaN or Inf.
1359 1BD2F 05           SETDEC                Digit counters are decimal.
1360 1BD31 AF2          C=0    W              (C(S) is flag for -0.)
1361 1BD34 15F3         C=DAT1 4              C(A)= #digits specified.
1362 1BD38 AF7          D=C    W              D= total #digits in field;
```

```
1363 1BD3B 173          D1=D1+ 4              D(S)= flag for -0.
1364 1BD3E 15F3         C=DAT1 4            C= #digits before radix.
1365 1BD42 959          ?B=0   M            Check for significant 0 later?
1366 1BD45 A0           GOYES  USnm07       No.
1367 1BD47 8A7          ?C#D   A            All integer field?
1368 1BD4A 50           GOYES  USnm07       No. Decimal digits, too.
1369 1BD4C A4E          C=C-1  S            Yes. For unit's digit check.
1370 1BD4F 1C3  USnm07  D1=D1- 4            D1 back to total #digit storage.
1371 1BD52 949          ?B=0   S            Negative number?
1372 1BD55 A1           GOYES  USnm15       No.
1373 1BD57 870          ?ST=1  s"S,M"       Yes. Sign specified?
1374 1BD5A 81           GOYES  USnm17       Yes.
1375 1BD5C CE           C=C-1  A            One digit posn for sign.
1376 1BD5E 4B0          GOC    USnm11       CRY= maybe -0.
1377 1BD61 CF           D=D-1  A            One less total digit position.
1378 1BD63 8AF          ?D#0   A            Only one total digit posn?
1379 1BD66 C0           GOYES  USnm17       No. s"S,M"=0 for "neg expr".
1380 1BD68 E7           D=D+1  A            Yes. Will need this posn for -0.
1381 1BD6A E6   USnm11  C=C+1  A            Don't use posn for sign, if -0.
1382 1BD6C A4F          D=D-1  S            Flag: check for -0 later.
1383 1BD6F 850  USnm15  ST=1   s"S,M"       Either sign specfd, or pos expr.
1384            *                           Now C=#digits before radix.
1385 1BD72 872  USnm17 ?ST=1   sNUME        Exponent specified?
1386 1BD75 F1           GOYES  USnm33       Yes. No leading zeros.
1387            *
1388 1BD77 7A95         GOSUB  ExpEXP       Expand Exponent to 5 digit form.
1389            *
1390            *          At this point, C(A)= #digits before radix,
1391            *                         A(A)= exponent+1
1392            *
1393 1BD7B E2           C=C-A  A            #zeros till first nonzero digit.
1394 1BD7D 5C0          GONC   USnm25       NC= sufficient digits.
1395 1BD80 95C          ?A#0   M            If negative exponent, OK.
1396 1BD83 70           GOYES  USnm25
1397 1BD85 D2           C=0    A            Insufficient digits...
1398 1BD87 A4F          D=D-1  S              unless -0 acceptable.
1399            *
1400            *    Now compute the nibble for rounding:
1401            *       rounding nibble
1402            *         = 13-(exponent)-(#digits specified past radix)
1403            *         = 13+(#digits till radix)-(exponent)-(total #digits)
1404            *         = 13+(#zeros)-(total #digits)
1405            *
1406 1BD8A E3   USnm25 D=D-C  A            D= #nib to round past fixed rdx.
1407 1BD8C 570          GONC   USnm33       NC= round OK.
1408 1BD8F CB           C=C+D  A            C= #zeros= total #digits.
1409 1BD91 AD1          B=0    M            E.g.: ".DD";.0006 . Mantissa=0.
1410 1BD94 109  USnm33  R1=C                R1= #zeros.
1411 1BD97 D2           C=0    A
1412 1BD99 3141         LCHEX  14           Fixed radix at position 14.
1413 1BD9D EB           C=C-D  A            C= nibble to round.
1414 1BD9F 490          GOC    USnm39       Nibble beyond 14; no round (P=0)
1415 1BDA2 8F00         GOSBVL =DECP=C      "P=C 0   in decimal mode".
          000
1416 1BDA9 AF4  USnm39 A=B     W            Expression to A.
```

```
1417 1BDAC D9          C=B    A            Save exponent in C(X) for later.
1418 1BDAE 8E00        GOSUBL =RND-12      Round.
          00
1419 1BDB4 5A0         GONC   USnm41       NC= no MAXREAL overflow.
1420 1BDB7 E4          A=A+1  A            MAXREAL returned for round ovfl.
1421 1BDB9 BD4         ASR    M            Exponent=500; A(M)=0999..99
1422 1BDBC B54         A=A+1  M            A(M)=1000...00 . Now A=1E500.
1423           *
1424 1BDBF AF8  USnm41 B=A    W            B= rounded expression.
1425 1BDC2 DA          A=C    A            Old exponent to A.
1426 1BDC4 862         ?ST=0  sNUME        Exponent specified?
1427 1BDC7 03          GOYES  USnm47       No.
1428           *                           Yes. Now sNUME=1 will indicate
1429           *                             a positive exponent.
1430 1BDC9 D2          C=0    A            "No zeros" to R1.
1431 1BDCB 959         ?B=0   M            Expression = 0?
1432 1BDCE 84          GOYES  USnm55       Yes. Only output "00..00E+000"
1433           *                          (Otherwise "5DE";0 gives 00000E-004)
1434 1BDD0 129         CR1EX               C= #digits before radix.
1435 1BDD3 7E35        GOSUB  ExpEXP        Expand exponent to 5 digit form.
1436 1BDD7 EA          A=A-C  A            A= exponent for display.
1437 1BDD9 958         ?A=0   M            Neg exponent for display?
1438 1BDDC 70          GOYES  USnm43       No.
1439 1BDDE 842         ST=0   sNUME        Yes. Flag: neg exp.
1440 1BDE1 F8          A=-A   A            Store in positive form.
1441 1BDE3 AB8  USnm43 B=A    X            B= expression with full exponent.
1442 1BDE6 D2          C=0    A
1443 1BDE8 A3E         C=C-1  X            00999 = max exponent display.
1444 1BDEB 8BA         ?A<=C  A            Exponent overflow?
1445 1BDEE 90          GOYES  USnm47       No.
1446 1BDF0 841  USGOVj ST=0   sFLOAT       Yes. Disable float after warning.
1447 1BDF3 68B4        GOTO   USGOVF       "USING Ovfl".
1448           *
1449 1BDF7 119  USnm47 C=R1                C= #zeros.
1450 1BDFA 95D         ?B#0   M            Rounded expr=0?
1451 1BDFD A0          GOYES  USnm51       No.
1452 1BDFF 94A         ?C=0   S            Yes. Now, check for unit's 0.
1453 1BE02 41          GOYES  USnm55       Not necessary.
1454 1BE04 5C0         GONC   USnm53       (BET) Force significant zero.
1455           *
1456 1BE07 94F  USnm51 ?D#0   S            Non-zero expr. -0 case?
1457 1BE0A 6E          GOYES  USGOVj       Yes; no space for sign.
1458 1BE0C 930         ?A=B   X            Exponent incremented in rounding?
1459 1BE0F 70          GOYES  USnm55       No.
1460 1BE11 CE   USnm53 C=C-1  A            Yes. One less zero.
1461 1BE13 4CD         GOC    USGOVj       CRY= insufficient digits.
1462 1BE16 D7   USnm55 D=C    A            #zeros to D(A) (for float check).
1463 1BE18 04          SETHEX
1464 1BE1A 7352        GOSUB  SAVE++       A=curr xqt addr, D1= AvMemEnd,
1465           *                             store #zeros in R0(9-5).
1466 1BE1E 3233        LCHEX  F33          C(0)= 3 for ASCII output,
          F
1467           *                             C(1)= flag for neg expr,
1468           *                             C(2)= flag for neg exponent.
1469 1BE23 94D         ?B#0   S            Neg expr?
```

```
1470 1BE26 50          GOYES  USnn57        Yes. C(1)=3.
1471 1BE28 BE6         CSR    B             No. C(1)=0.
1472 1BE2B 862  USnn57 ?ST=0  sNUME         Negative exponent?
1473 1BE2E 80          GOYES  USnn59        Yes. C(2)=F.
1474 1BE30 AA2         C=0    XS            Pos exp. C(2)=0.
1475 1BE33 842         ST=0   sNUME
1476 1BE36 145  USnn59 DAT1=C A             Store offset for digit.
1477 1BE39 1CF         D1=D1- 16            D1 to numeric expression.
1478 1BE3C AF9         C=B    W             Formatted expression.
1479 1BE3F 1557        DAT1=C W             Re-write formatted expr.
1480 1BE43 131         D1=A                 D1= execution address.
1481 1BE46 861         ?ST=0  sFLOAT        Floating symbols (D's) ?
1482 1BE49 C6          GOYES  IMx10j        No. Next symbol.
1483           *                          Fall into CHKFLT directly.
1484           *
```

```
1485                    EJECT
1486                ******************************************************
1487                *           Code from above falls into here!
1488                *
1489 1BE4B 843    CHKFLT ST=0   sZeros          Flag for "D not found yet."
1490 1BE4E 870           ?ST=1  s"S,M"          Sign specified, or pos expr?
1491 1BE51 40            GOYES  CHKFL5          Yes.
1492 1BE53 E7            D=D+1  A               No. Allow one more zero for sign.
1493 1BE55 101    CHKFL5 R1=A                   Save D1 for end of float check.
1494 1BE58 D1            B=0    A               To count floats.
1495                *
1496 1BE5A 7CEB   CHKFL7 GOSUB  =BYTscA
1497                *
1498 1BE5E 44            CON(2) \D\
1499 1BE60 B60           REL(3) FLOATd
1500                *
1501 1BE63 1D            CON(2) uMULT          Multiplier token.
1502 1BE65 A40           REL(3) CHKF11
1503                *
1504 1BE68 35            CON(2) \S\
1505 1BE6A A70           REL(3) FLTxsm
1506                *
1507 1BE6D 85            CON(2) \X\
1508 1BE6F 570           REL(3) FLTxsm
1509                *
1510 1BE72 D4            CON(2) \M\
1511 1BE74 070           REL(3) FLTxsm
1512                *
1513 1BE77 0D            CON(2) uSTRPT
1514 1BE79 860           REL(3) FLTstr
1515                *
1516 1BE7C E2            CON(2) \.\
1517 1BE7E 220           REL(3) CHKFL9
1518                *
1519 1BE81 34            CON(2) \C\
1520 1BE83 450           REL(3) FLOATc
1521                *
1522 1BE86 A5            CON(2) \Z\
1523 1BE88 810           REL(3) CHKFL9
1524                *
1525 1BE8B 05            CON(2) \P\
1526 1BE8D A40           REL(3) FLOATc
1527                *
1528 1BE90 25            CON(2) \R\
1529 1BE92 E00           REL(3) CHKFL9
1530                *
1531 1BE95 00            CON(2) 0
1532                *
1533 1BE97 316E          LC(2)  EndNum         Any token>=Endnum ends num field.
1534 1BE9B 9E2           ?A<C   B              End of numeric field?
1535 1BE9E CB            GOYES  CHKFL7         No. Keep checking tokens.
1536 1BEA0 852    CHKFL9 ST=1   sNUME          Flag: End of float check.
1537 1BEA3 05            SETDEC
1538 1BEA5 870           ?ST=1  s"S,M"          Neg expr with implied sign?
1539 1BEA8 B4            GOYES  ENDFLT         No. Output floats.
```

```
1540 1BEAA CD          B=B-1  A        Yes. Recover one posn for sign.
1541 1BEAC 564         GONC   ENDFLT    (BET) Output floats.
1542           *
1543 1BEAF 1C7  CHKF11 D1=D1- 8        Multiplier: skip over
1544 1BEB2 57A         GONC   CHKFL7    (BET)       decimal digits.
1545           *
1546           *
1547 1BEB5 603C IMx10j GOTO   IMxq10    Execute next symbol.
1548           *
```

```
1549                    EJECT
1550              ********************************************************
1551              *
1552 1BEB9          CHKFLA                          (From SKPFLT.)
1553 1BEB9 171             D1=D1+ 2                Don't skip current char.
1554 1BEBC 133             AD1EX                   Put D1 in A.
1555 1BEBF 131             D1=A
1556 1BEC2 76B1            GOSUB  ROu2CA           RO(9-5)=#zeros to C(A).
1557 1BEC6 D7              D=C    A                #zeros to D(A).
1558 1BEC8 5C8             GONC   CHKFL5           (BET) Count next block of floats.
1559              *
1560              ************************************************
1561              *
1562 1BECB 853    FLOATd ST=1    sZeros           Found a D!
1563 1BECE 7874            GOSUB  COUNTP           Count float positions.
1564 1BED2 4DC             GOC    CHKFL9           CRY= end of float check.
1565 1BED5 CD              B=B-1  A                Nullify next B+1.
1566              *
1567 1BED7 05     FLOATc SETDEC
1568 1BED9 E5              B=B+1  A                One more float posn.
1569 1BEDB 04              SETHEX
1570 1BEDD 6C7F   FLOTd5 GOTO    CHKFL7           Keep looking for floats.
1571              *
```

```
1572                    EJECT
1573              ******************************************************
1574              *
1575 1BEE1 1CB    FLIstr D1=D1- 12            Skip past string pointers.
1576 1BEE4 863    FLIxsm ?ST=0  sZeros        Found a D yet?
1577 1BEE7 6F             GOYES  FLOTd5       No. Keep checking for floats.
1578 1BEE9 870            ?ST=1  s"S,M"       Implied neg sign?
1579 1BEEC 70             GOYES  ENDFLT       No. Output floats.
1580 1BEEE 8AB            ?D=0   A            Yes. Down to sign?
1581 1BEF1 CE             GOYES  FLOTd5       Yes. Keep checking for floats.
1582              *                           No. Fall into ENDFLT.
1583              *
1584 1BEF3 DB     ENDFLT C=D     A            New #zeros to C.
1585 1BEF5 7984           GOSUB  STORB        #zeros to R0(9-5), store B in R3.
1586 1BEF9 7494           GOSUB  STORB1       R3 shift left 5 times, B to R3(A).
1587              *
1588 1BEFD 7E24   ENDFL7  GOSUB  CNTOUT       Count floats.
1589 1BF01 43B            GOC    IMx10j       CRY= end of float output.
1590 1BF04 855            ST=1   sRturn       Return from display.
1591 1BF07 70EC           GOSUB  USGch4       Display blank.
1592 1BF0B 51F            GONC   ENDFL7       (BET)  Next float output.
1593              *
```

```
1594                    EJECT
1595                ******************************
1596                *
1597 1BF0E 24    USG"D" P=      4              Sets R0(S)=4 for digit code.
1598 1BF10 861         ?ST=0  sFLOAT         Out of the floats?
1599 1BF13 41          GOYES  USGd03          Yes.
1600                *
1601 1BF15 7614  SKPFLT GOSUB  CNTOUT         Count output chars.
1602 1BF19 5B9         GONC   IM*10j          Keep skipping floats.
1603 1BF1C 862         ?ST=0  sNUME          Through with floats?
1604 1BF1F A9          GOYES  CHKFLA          No. Check floats again.
1605                *
1606 1BF21 842         ST=0   sNUME          (Off flag: Through with floats.)
1607 1BF24 841         ST=0   sFLOAT         "Out of the floats."
1608 1BF27 870   USGd03 ?ST=1  s"S,M"         Display digit?
1609 1BF2A 90          GOYES  USG"*"          (BET) Yes. (CRY set for USGz*7)
1610                *                        No. Display implied neg sign.
1611                *
1612 1BF2C 21    USG"Z" P=      1                  To disp "0" if in the zeros.
1613                *                       Test: In the floats?  (Only for
1614 1BF2E 871         ?ST=1  sFLOAT        one's digit Z or first decimal D.)
1615 1BF31 4E          GOYES  SKPFLT          Yes. Skip floats.
1616 1BF33       USG"*"
1617 1BF33 118         C=R0                   Save P in R0(S),
1618 1BF36 80FF        CPEX   15                 C(9-5)=#zeros.
1619 1BF3A 108         R0=C
1620 1BF3D 470         GOC    USGz*7          CRY set for "D" symbol.
1621                *                        Sign output already,
1622 1BF40 860   USGz*3 ?ST=0  s"S,M"           or specified in field?
1623 1BF43 64          GOYES  USG"S"          No. Neg sign takes digit's place.
1624                *
1625 1BF45 863   USGz*7 ?ST=0  sZeros         In the zeros?
1626 1BF48 76          GOYES  USGdgt          No. Display digit.
1627 1BF4A 7131        GOSUB  CSRW9j          C(A)= #zeros.
1628 1BF4E 843         ST=0   sZeros         In case end of zeros...
1629 1BF51 05          SETDEC
1630 1BF53 CE          C=C-1  A              Decrement zero count.
1631 1BF55 04          SETHEX
1632 1BF57 48E         GOC    USGz*3         No more zeros. Display digit,
1633                *                          or floating implied neg sign.
1634 1BF5A 853         ST=1   sZeros         Nope, not end of zeros.
1635 1BF5D 75BA        GOSUB  =CSL2R0        Store counter back in R0(9-5).
1636 1BF61 80DF        P=C    15             Restore "Z" or "*" flag.
1637 1BF65 6D8C  USGc0j GOTO   USGchO         Display " " or "*" or "0".
1638                *
```

```
1639                    EJECT
1640              ********************************************************
1641              *
1642 1BF69 21     USG"C" P=      1
1643 1BF6B        USG"P"
1644 1BF6B 863           ?ST=0  sZeros          In the zeros?
1645 1BF6E 71            GOYES  USGcp3          No. Disp "," or ".".
1646 1BF70 871           ?ST=1  sFLOAT          In the floats?
1647 1BF73 2A            GOYES  SKPFLT          Yes. Skip floats.
1648              *                             No. Check which symbol.
1649 1BF75 118           C=RO                   Numeric symbol code to C(S).
1650 1BF78 80FF          CPEX   15              Code to P.
1651 1BF7C 881           ?P#    1               "Z"?
1652 1BF7F 6E            GOYES  USGc0j          No. Display " " or "*".
1653 1BF81 80FF          CPEX   15              Yes. Display "," or ".".
1654 1BF85 636C  USGcp3  GOTO   USGch5
1655              *
1656              ********************************************************
1657              *
1658 1BF89 2E     USG"S" P=      14              Flag for "S" symbol.
1659 1BF8B        USG"M"                         (P=0)
1660 1BF8B 76E0          GOSUB  StAVE+          No. D1 to A, set D1=AvMemEnd.
1661 1BF8F 147           C=DAT1 A               Read flags: C(1)=expr sign,
1662              *                               C(2)= exponent sign.
1663 1BF92 871           ?ST=1  sFLOAT          In floats?
1664 1BF95 70            GOYES  USGsm1          Yes. sNUME not "exponent".
1665 1BF97 872           ?ST=1  sNUME           Exponent being output?
1666 1BF9A 40            GOYES  USGsm3          Yes.
1667 1BF9C F2    USGsm1  CSL    A               No. Expr sign to C(XS).
1668 1BF9E 133   USGsm3  AD1EX                  Restore D1 from A.
1669 1BFA1 92A           ?C=0   XS              Positive?
1670 1BFA4 70            GOYES  USGsm5          Yes.
1671 1BFA6 2F            P=     15              No. Output "-" positively.
1672 1BFA8 850           ST=1   s"S,M"          "Sign has been output."
1673 1BFAB 6F3C  USGsm5  GOTO   USGch4          Output "+" or " ".
1674              *
```

```
1675                    EJECT
1676              ************************************************
1677              *
1678 1BFAF 7012  USGdgt GOSUB  AVEDT1          A=curr xqt addr, D1=AvMemEnd,
1679              *                             C(W)= entire numeric expr.
1680 1BFB3 BF2          CSL    W               Next mantissa digit for output.
1681 1BFB6 23           P=     3               To protect exponent digits.
1682 1BFB8 B96          CSR    WP              Shift exponent digits back.
1683 1BFBB 862          ?ST=0  sNUME           Exponent being output?
1684 1BFBE 70           GOYES  USGdg1          No.
1685 1BFC0 303          LCHEX  3               Yes.
1686 1BFC3 F2           CSL    A               Next exponent digit for output.
1687 1BFC5 1557  USGdg1 DAT1=C W               Replace shifted expr.
1688 1BFC9 550          GONC   USGdg3          NC only for mantissa digits.
1689 1BFCC 17B          D1=D1+ 12              To ASCII mantissa digit-3.
1690 1BFCF 172   USGdg3 D1=D1+ 3               To ASCII exponent digit.
1691 1BFD2 D6           C=A    A               Xqt addr to C.
1692 1BFD4 137   USGdg7 CD1EX                  D1=xqt addr, C=ASCII.
1693 1BFD7 6B3C         GOTO   USGch*          Output ASCII.
1694              *
```

```
1695                    EJECT
1696                ********************************************
1697                *
1698 1BFDB 7220   USG"B" GOSUB  POPTS+          Test stack item.
1699 1BFDF 443          GOC    USer1k          CRY=non-numeric. "Invalid USING".
1700 1BFE2 8E00         GOSUBL =FLTDH
          00
1701 1BFE8 4D0          GOC    USGB03          CRY= in range.
1702 1BFEB 831          ?XM=0                  Negative?
1703 1BFEE 80           GOYES  USGB03          Yes.
1704 1BFF0 8C00         GOLONG =InvArg         Out of range. "Invalid Arg".
          00
1705                *
1706 1BFF6 7C20   USGB03 GOSUB  NwOFF+          Write back out hex expression.
1707 1BFFA 1C4          D1=D1- 5               D1= addr of hex expr.
1708 1BFFD 66DF         GOTO   USGdg7          Output the B char.
1709                *
1710                ************************************
1711                *
1712 1C001 7820   POPTS+ GOSUB  NwOFFS          New offset to AvMemEnd field.
1713 1C005 135    POPTST D1=C                   D1= start of stack item.
1714 1C008 1537         A=DAT1 W               Read item.
1715 1C00C 309          LCHEX  9
1716 1C00F B02          C=C-A  P               Numeric?
1717 1C012 01           RTN                    CRY set: not numeric.
1718                *                          CRY clear: numeric.
1719                *
1720                ********************************************
1721                *
1722 1C014 65D0   USer1k GOTO   USer1j          From USG"B" above.
1723                *
1724                ************************************************
1725                *
1726 1C018 133    DT1C-A AD1EX
1727 1C01B 24           P=     4
1728 1C01D 131    DT1C.A D1=A
1729 1C020 E2           C=C-A  A
1730 1C022 674A         GOTO   =BldIM+
1731                *
```

```
1732                    EJECT
1733                ****************************************************************
1734                ****************************************************************
1735                **
1736                ** Name:(S) NWOFFS  -  Recover old offset, store new one in RAM
1737                **
1738                ** Category:   GENUTL
1739                **
1740                ** Purpose:
1741                **      Recover old offset from AVMEMEND, store a new one
1742                **      in the same location.  (Utility for IMAGE execution,
1743                **      but can be used anywhere.)
1744                **
1745                ** Entry:
1746                **      D1=address+5 for which new offset will be computed
1747                **      Old offset resides at AVMEMEND
1748                **
1749                ** Exit:
1750                **      Carry clear
1751                **      New offset stored in AVMEMEND
1752                **      C(A)=recovered offset from AVMEMEND (recovered means
1753                **        that the addition has been performed on the offset
1754                **        to recover the address)
1755                **      D1=A(A)=AVMEMEND+5
1756                **
1757                ** Calls:      STAVE+ (SETAVE), CA2D1+
1758                **
1759                ** Uses.......
1760                **  Exclusive: A(A),C(A),D1
1761                **  Inclusive: A(A),C(A),D1
1762                **
1763                ** Stk lvls:   1
1764                **
1765                ** Detail:
1766                **     =NWOFFS D1=D1- 5
1767                **             AD1EX
1768                **             GOSBVL =SETAVE      Set D1=C=AVMEMEND
1769                **             A=A-C  A            Compute new offset.
1770                **             C=DAT1 A            Fetch old offset.
1771                **             DAT1=A A            Store new offset.
1772                **             GOTO   CA2D1+       Recover compute address.
1773                **
1774                ** History:
1775                **
1776                **    Date     Programmer              Modification
1777                **  --------   ----------   ----------------------------------
1778                **  12/08/82   MB           Documentation
1779                **
1780                ****************************************************************
1781                ****************************************************************
1782                *
1783 1C026 76D9 =NWOFF+ GOSUB  =SETAVE         Set D1= AVMEMEND.
1784 1C02A 5B0         GONC   NWOFF3         (BET)
1785                *
1786 1C02D 1C4  =NWOFFS D1=D1- 5
```

```
1787 1C030 7140          GOSUB  StAVE+        A=curr xqt addr, D1= AvMemEnd.
1788 1C034 EA            A=A-C  A
1789 1C036 147   NwOFF3  C=DAT1 A
1790 1C039 141           DAT1=A A
1791 1C03C 6910          GOTO   CA2D1+
1792             *
```

```
1793                    EJECT
1794            ********************************************************************
1795            ********************************************************************
1796            **
1797            ** Name:(S) ENDIMG  -  Process end of IMAGE string
1798            **
1799            ** Category:   EXCUTL
1800            **
1801            ** Purpose:
1802            **      Process uIMend token at end of IMAGE string.
1803            **
1804            ** Entry:
1805            **      P      =  0
1806            **      RAM storage as shown in IMxqt header.
1807            **
1808            ** Exit:
1809            **      If "not ouput field found" (S10=0), generates
1810            **         an "Invalid USING" error.
1811            **      Else:
1812            **         P=0
1813            **         D1=AvMemEnd+5
1814            **         C(A)=address of start of IMAGE string.  If there
1815            **           are more output fields, the IMAGE string can now
1816            **           be recycled.
1817            **         S0=0,S1=0,S2=0,S2=0,S6=0
1818            **
1819            ** Calls:      GETSTA, CLOST+, RCVOFS
1820            **
1821            ** Uses.......
1822            **  Exclusive: D1
1823            **  Inclusive: D1,A(A),C(A),D(A),S0,S1,S2,S3,S6
1824            **
1825            ** Stk lvls:   2
1826            **
1827            ** NOTE:
1828            **      During IMAGE execution (output or enter), when the end-
1829            **      of-image is encountered the routine TstEnd should be
1830            **      called to determine if at the end of the output list
1831            **      (or enter list).  If so, exit to NXTSTM.  If not, call
1832            **      IMGEND to recycle the image string.
1833            **
1834            ** Detail:
1835            **     =ENDIMG GOSUB  GETSTA          Get status bits from RAM
1836            **             GOSUB  =CLOST+         Set S0,S1,S2,S3,S6=0
1837            **             ?ST=0  10              Output field found?
1838            **             GOYES  <Invalid USING error>   No. Error.
1839            **             D1=D1+ 8               Gives D1+3 in RCVOFS
1840            **          ... fall into RCVOFS...   Recover offset to start
1841            **                                      of image string.
1842            **
1843            ** History:
1844            **
1845            **    Date     Programmer             Modification
1846            **  --------   ----------   ----------------------------------
1847            **  12/08/82   MB           Documentation
```

```
1848            **
1849            ********************************************************************
1850            ********************************************************************
1851            *
1852            ********************************************************************
1853            ********************************************************************
1854            **
1855            ** Name:(S) RCVOFS  -  Recover offset from RAM storage
1856            ** Name:(S) C+A2D1  -  Recover offset from RAM storage
1857            ** Name:    CA2D1+  -  Recompute offset from RAM storage
1858            **
1859            ** Category:   GENUTL
1860            **
1861            ** Purpose:
1862            **      To recover a 5-nibble offset from RAM (recover means
1863            **      to fetch the offset, perform addition to recompute
1864            **      the original address).
1865            **
1866            ** Entry:
1867            **      RCVOFS: offset to recover resides at D1-5
1868            **      C+A2D1: offset to recover resides at D1
1869            **
1870            ** Exit:
1871            **      Carry clear
1872            **      D1=A(A)=address+5 where offset was found
1873            **      C(A)=recovered offset (offset was added to D1 to
1874            **        recompute old address)
1875            **
1876            ** Calls:      none
1877            **
1878            ** Uses.......
1879            **  Exclusive: A(A),C(A)
1880            **             C+A2D1 also uses D1 (does a D1+5)
1881            **
1882            ** Stk lvls:   0
1883            **
1884            ** Detail:
1885            **     =RCVOFS D1=D1- 5
1886            **     =C+A2D1 C=DAT1 A
1887            **      CA2D1+ D1=D1+ 5
1888            **             AD1EX
1889            **             D1=A
1890            **             C=A+C  A
1891            **             RTNCC
1892            **
1893            ** History:
1894            **
1895            **    Date     Programmer              Modification
1896            **  --------   ----------   ----------------------------------
1897            **  12/08/82   MB           Documentation
1898            **
1899            ********************************************************************
1900            ********************************************************************
1901            *
1902 1C040     =ENDIMG                      End of image string.
```

```
1903 1C040 7F10          GOSUB  GETSTA          Get status bits from storage.
1904 1C044 7289          GOSUB  =CLOST+         Clear 0 nib of status, sSpecl.
1905 1C048 86A           ?ST=0  sFOUND          Output fields found?
1906 1C04B 9C            GOYES  USer1k          No. "Invalid USING".
1907 1C04D 177           D1=D1+ 8               Gives D1+3, points to offset
1908               *                              to start of IMAGE string.
1909               *                            Fall through to RCVOFS...
1910               *
1911 1C050 1C4   =RCVOFS D1=D1- 5               Recover offset from RAM,
1912               *                              compute address.
1913 1C053 147   =C+A2D1 C=DAT1 A
1914 1C056 174    CA2D1+ D1=D1+ 5
1915 1C059 133    CA2D1. AD1EX
1916 1C05C 131           D1=A
1917 1C05F C2            C=A+C  A
1918 1C061 03            RTNCC
1919               *
1920               ********************************************
1921               *
1922 1C063 7399   GETSTA GOSUB  =SetAVM
1923 1C067 174    GETST1 D1=D1+ 5
1924 1C06A 147           C=DAT1 A
1925 1C06D 0A            ST=C
1926 1C06F 03            RTNCC
1927               *
1928               ********************************************
1929 1C071 71A9   SAVE++ GOSUB  =CSL2RO
1930 1C075 133    StAVE+ AD1EX
1931 1C078 6789   Setave GOTO   =SetAVE
1932               *
1933               ********************************************
1934               *
1935 1C07C 118    ROu2CA C=RO
1936 1C07F 8D00 =CSRW9j GOVLNG =CSRWP9
           000
1937               *
```

```
1938                    EJECT
1939            ********************************************************************
1940            ********************************************************************
1941            **
1942            ** Name:(S) GetEXP  -  Expression execute for IMAGE output list
1943            **
1944            ** Category:   EXCUTL
1945            **
1946            ** Purpose:
1947            **      Call EXPEXC for items in IMAGE output list, screen
1948            **      expression for valid type.
1949            **
1950            ** Entry:
1951            **      P      = 0
1952            **      R3(A)=Program Counter
1953            **      RAM storage as shown in IMGxqt header.
1954            **      S3 and S6 determine valid expression types:
1955            **         S6=1 means "numeric expression acceptable"
1956            **         S3=0 means "string expression acceptable"
1957            **         S3#S6 means "complex acceptable"
1958            **           valid type     S3  S6
1959            **           ----------     --  --
1960            **           numeric         1   1
1961            **           string          0   0
1962            **           complex         1   0
1963            **           any (K or H)    0   1
1964            **
1965            ** Exit:
1966            **      If expression is not of valid type, "Invalid USING".
1967            **      Else:
1968            **         P=0
1969            **         Carry clear
1970            **         S6=0
1971            **         If numeric or complex expression:
1972            **           RES register has been updated
1973            **           A(W)=numeric expression (or =real part, in
1974            **             the case of complex)
1975            **         If string expression, A(W)=string header except
1976            **           that A(B)=00.
1977            **
1978            ** Calls:      TstEnd, NXTEXP, CKINFO, POPMTH, AVE=D1, GETST1,
1979            **             POPTST, PUTRES
1980            **
1981            ** Uses.......
1982            **      Calls EXPEXC, which may use anything.
1983            **
1984            ** Stk lvls:   Calls EXPEXC, which may use anything.
1985            **             5 levels available to EXPEXC.
1986            **
1987            ** Algorithm:
1988            **      Test output list for end-of-list.  If so, to NXTSTM.
1989            **      Call NXTEXP, which stores status bits and offset
1990            **        to D1 in RAM, jumps to EXPEXC.
1991            **      Pop math stack.
1992            **      Restore status bits from RAM.
```

```
1993            **      If numeric expression:
1994            **  2)     If S6=1, then go to 4). Else go to 3).
1995            **      If string expression:
1996            **         If S3=0, then return. Else go to 3).
1997            **      If complex expression:
1998            **         If S3=0, then go to 2).
1999            **         Else (S3=1) if S6=0 then go to 4).
2000            **  3)  Exit to "Invalid USING" error.
2001            **  4)  Put expression in RES register. Return.
2002            **
2003            ** History:
2004            **
2005            **    Date     Programmer              Modification
2006            **  --------   ----------   ----------------------------------
2007            **  12/08/82   MB           Documentation
2008            **
2009            ********************************************************************
2010            ********************************************************************
2011            *
2012 1C086 7570 =GetEXP GOSUB  TstEnd
2013 1C08A 596          GONC   ENDSTj          End of stmt. CR,LF and Next stmt.
2014 1C08D 7662         GOSUB  NXTEXP          Get next expression.
2015 1C091 8E00         GOSUBL =CKINFO         Verify output device info.
           00
2016 1C097 10B          R3=C
2017 1C09A 133          AD1EX
2018 1C09D 131          D1=A
2019 1C0A0 D8           B=A    A
2020 1C0A2 8E00         GOSUBL =POPMTH
           00
2021 1C0A8 8E00         GOSUBL =AVE=D1         Set AvMemEnd=D1.
           00
2022 1C0AE 75BF         GOSUB  GETST1          Get status bits from storage.
2023 1C0B2 D9           C=B    A               C= addr of expression.
2024            *
2025            *  Now test expression for valid type:
2026            *          At entry, sSpec1=1 means "numeric acceptable".
2027            *                    sZeros=0 means "string acceptable".
2028            *             sSpec1 # sZeros means "complex acceptable".
2029            *
2030            *                                sSpec1  sZeros
2031            *          Numeric field:          1       1
2032            *          String field:           0       0
2033            *          Complex field:          0       1
2034            *          K or H field:           1       0
2035            *
2036 1C0B4 7D4F         GOSUB  POPTST          Test stack item.
2037 1C0B8 421          GOC    GtEX05          CRY= non-numeric.
2038 1C0BB 866   GtEX01 ?ST=0  sSpec1          Numeric acceptable?
2039 1C0BE C2           GOYES  USer1j          No. "Invalid USING".
2040 1C0C0 8E00  GtEX03 GOSUBL =PUTRES         Expr to RES register.
           00
2041 1C0C6 846   GtEX04 ST=0   sSpec1          Clear special flag.
2042 1C0C9 03           RTNCC
2043            *
```

```
2044 1C0CB B04   GtEX05 A=A+1  P
2045 1C0CE 968          ?A=0   B              String expr?
2046 1C0D1 41           GOYES  GtEX09         Yes.
2047 1C0D3 B04          A=A+1  P              No.
2048 1C0D6 96C          ?A#0   B              Complex expr?
2049 1C0D9 11           GOYES  USer1j         No. "Invalid USING".
2050 1C0DB 863          ?ST=0  sZeros         Yes. K or H field?
2051 1C0DE DD           GOYES  GtEX01         Possibly.
2052 1C0E0 866          ?ST=0  sSpec1         No. Complex field?
2053 1C0E3 DD           GOYES  GtEX03         Yes.
2054 1C0E5 863   GtEX09 ?ST=0  sZeros         (From GtEX05) String acceptable?
2055 1C0E8 ED           GOYES  GtEX04         Yes.
2056            *
2057            *
2058 1C0EA 8CCF  USer1j GOLONG =IvUSGj        No. "Invalid USING".
         3F
2059            *
2060            **********************************************
2061            *
2062 1C0F0 7B00  USGend GOSUB  TstEnd         If eol, then exit.
2063 1C0F4 534   ENDSTj GONC   ENDSTM         End of stmt.
2064 1C0F7 754F         GOSUB  ENDIMG         Check for recycling of image.
2065 1C0FB 60AA         GOTO   IMxq27         Yes; start at beginning of img.
2066            *
```

```
2067                    EJECT
2068            ****************************************************************
2069            ****************************************************************
2070            **
2071            ** Name:(S) TstEnd  -  Test IMAGE output list for end of list
2072            **
2073            ** Category:   EXCUTL
2074            **
2075            ** Purpose:
2076            **      Test IMAGE output list for end-of-list.  If not,
2077            **      positions D0 to next expression.
2078            **
2079            ** Entry:
2080            **      P       =  0
2081            **      R3(A)=Program Counter
2082            **      RAM storage as shown in IMGxqt header
2083            **
2084            ** Exit:
2085            **      P       =  0
2086            **      Carry clear: end of output list (D0 points past EOL,
2087            **           "@" or "!")
2088            **         D1 points to first image token
2089            **         A(B)=first image token
2090            **         C(B)=ASCII "#" for test of first image token.
2091            **      Carry set: D0 points to next expression in output list
2092            **
2093            ** Calls:       EOLXCK
2094            **              If end-of-list, also calls: SetAVE, C+A2D1
2095            **
2096            ** Uses.......
2097            **  Exclusive: A(B),C(W),D0,D1
2098            **  Inclusive: A(B),C(W),D0,D1
2099            **
2100            ** Stk lvls:   1
2101            **
2102            ** NOTE:
2103            **      If end-of-list, A(B) and C(B) are ready to test
2104            **      first image token for "#".  If the first token
2105            **      is a "#", then a CR-LF should not be sent out.
2106            **
2107            ** Algorithm:
2108            **      Fetch Program Counter from R3(A), copy to D0.
2109            **   1) Read byte from DAT0.
2110            **      If A(B) not "," or ";" then return carry set
2111            **        (A(B) must be first byte in expression)
2112            **      Increment D0+2
2113            **      Test A(B) for EOL, "@" or "!".  If no match,
2114            **        go to 1) (must be another "," or ";")
2115            **      (Match with EOL, "@" or "!"):
2116            **      Recover offset to start of IMAGE string,
2117            **        put address in D1.
2118            **      Read first image token into A(B).
2119            **      Load ASCII "#" into C(B).
2120            **      Return carry clear.
2121            **
```

```
2122                    ** History:
2123                    **
2124                    **    Date     Programmer          Modification
2125                    **  --------   ----------  ----------------------------------
2126                    **  12/08/82   MB          Documentation
2127                    **
2128                    ************************************************************
2129                    ************************************************************
2130                    *
2131 1C0FF 11B  =TstEnd C=R3
2132 1C102 134         DO=C
2133 1C105 14A   TstEn3 A=DAT0 B
2134 1C108 3100        LC(2)  =tEOL          Test for "," or ";".
2135 1C10C 9E2         ?A<C   B              Comma or semicolon?
2136 1C10F 00          RTNYES                No. Expression.
2137                    *       This test allows the command  ' DISP USING " "; '.
2138                    *       Otherwise, the semicolon with no output items would
2139                    *         cause an error.  Also allows two or more adjacent
2140                    *         commas or semicolons between output items (this is
2141                    *         different from HP-75), or output stream to end
2142                    *         in a comma or semicolon.
2143                    *
2144 1C111 161           DO=DO+ 2              DO to next expr.
2145 1C114 8F00        GOSBVL =EOLXCK          Sets carry if EOL, @ or ! .
           000
2146 1C11B 59E         GONC   TstEn3         NC= either expr or "," or ";" .
2147 1C11E 7ED8        GOSUB  =SetAVE
2148 1C122 177         D1=D1+ 8
2149 1C125 7A2F        GOSUB  C+A2D1
2150 1C129 135         D1=C
2151 1C12C 3132        LCASC  \#\            Ready for "#" test.
2152 1C130 1C1   TstEn5 D1=D1- 2              To first BldIMG char.
2153 1C133 14B         A=DAT1 B              Read first BldIMG char.
2154 1C136 03          RTNCC                 Alert caller to end of stmt.
2155                    *
2156 1C138 962   ENDSTM ?A=C   B              # symbol? (suppress CR-LF?)
2157 1C13B 80          GOYES  NXTstm         Yes.
2158 1C13D 8E00        GOSUBL =SENDEL        No. Send out ENDLINE.
           00
2159 1C143 20    NXTstm P=     0
2160 1C145 8C00        GOLONG =PART3         To DPART3, to next stmt.
           00
2161                    *
```

```
2162                    EJECT
2163            ****************************************************************
2164            ****************************************************************
2165            **
2166            ** Name:(S) USloop  -  Loop on IMAGE multiplier
2167            **
2168            ** Category:   EXCUTL
2169            **
2170            ** Purpose:
2171            **      To process a loop-on-multiplier token while executing
2172            **      an IMAGE statement.  Repositions D1 back to start
2173            **      of multiplier loop.
2174            **
2175            ** Entry:
2176            **   For a fixed jump (jump back a fixed number of nibbles),
2177            **      P=#nibbes-1 to jump
2178            **         P=3 for uLOOPB (loop on byte -- 4 nibble jump)
2179            **         P=15 for uLOOPS (loop on string -- 16 nib jump)
2180            **   For a jump whose length is calculated by a 5-nibble
2181            **      field,
2182            **         P=0 for uLOOPP (loop on parentheses)
2183            **   D1=address of loop token in BldIMG stream
2184            **
2185            ** Exit:
2186            **      Carry clear
2187            **      (P unchanged)
2188            **      If multiplier has not expired:
2189            **        loop counter has been decremented.
2190            **        D1 points to start of multiplier loop.
2191            **      If multiplier has expired:
2192            **        the reference counter has been copied into the
2193            **          loop counter.
2194            **        D1 is left as it was passed (points to loop token).
2195            **
2196            ** Calls:      CK"ON"
2197            **
2198            ** Uses.......
2199            **  Exclusive: C(A),D(A),D1
2200            **  Inclusive: C(A),D(A),D1,....
2201            **
2202            ** Stk lvls:   1
2203            **
2204            ** NOTE:
2205            **      USloop checks if the ATTN key has been hit; if so, it
2206            **      exits through PART3 (output handler), which goes to
2207            **      NXTSTM.  Thus, an image string like "9999X" will allow
2208            **      the user to abort it with the ATTN key.
2209            **
2210            ** Algorithm:
2211            **      Copy D1 to D(A).
2212            **      Check ATTN key; if pressed, exit.
2213            **      Increment D1 by P+1.
2214            **      If P#0 (loop on byte or string), go to 2)
2215            **      Else (loop on parentheses):
2216            **        Move D1 to offset storage
```

```
2217            **        Recover offset to start of loop
2218            **    2) Decrement loop counter
2219            **       If counter not expired, return.
2220            **       Else (counter expired):
2221            **         Copy reference counter to loop counter.
2222            **         Restore D1 from D(A), return.
2223            **
2224            ** History:
2225            **
2226            **     Date      Programmer              Modification
2227            **   --------    ----------   ----------------------------------
2228            **   12/08/82    MB           Documentation
2229            **
2230            ********************************************************************
2231            ********************************************************************
2232            *
2233 1C14B 137  =USloop CD1EX                    Save D1.
2234 1C14E 8F00         GOSBVL =CK"ON"          Check for ATTN key.
           000
2235 1C155 5DE          GONC   NXTstm           NC= ATTN hit. To NXTSTM.
2236 1C158 D7           D=C    A                Save D1=current xqt addr.
2237 1C15A 809          C+P+1                   Loop back to multiplier addr.
2238 1C15D 135          D1=C                    D1= multiplier address.
2239 1C160 880          ?P#    0                Loop on parentheses?
2240 1C163 B2           GOYES  DCRM05           No, loop on byte or string.
2241 1C165 1C5          D1=D1- 6                Yes. -1 for C+P+1, -5 to offset.
2242 1C168 77EE         GOSUB  C+A2D1           Recover offset to open paren addr.
2243 1C16C 1C6          D1=D1- 7                D1= resume addr if ctr expires.
2244 1C16F 137          CD1EX                   Now D1= start of paren loop.
2245 1C172 D7           D=C    A                Save D1= current xqt addr.
2246 1C174 591          GONC   DCRM05           (BET) Execute multiplier.
2247            *
```

```
2248                    EJECT
2249            ****************************************************************
2250            ****************************************************************
2251            **
2252            ** Name:(S) DCRMNT  -  Decrement multiplier in IMAGE string
2253            **
2254            ** Category:   EXCUTL
2255            **
2256            ** Purpose:
2257            **      To decrement loop counter in IMAGE string.  An image
2258            **      symbol with a multiplier causes a loop which must
2259            **      decrement the counter each time.
2260            **
2261            ** Entry:
2262            **      P       = 0
2263            **      D1 points to uMULT token (multiplier)
2264            **
2265            ** Exit:
2266            **      P       = 0
2267            **      Carry clear
2268            **      D1 ponts to next executing token (D1-8 from entry)
2269            **      Loop counter has been decremented.
2270            **      If an open parentheses loop, see note below.
2271            **
2272            ** Calls:      none
2273            **
2274            ** Uses.......
2275            **  Exclusive: A(B),C(A),D1
2276            **
2277            ** Stk lvls:   0
2278            **
2279            ** NOTE:
2280            **      If the loop counter is for a parentheses loop which
2281            **      has not been closed yet (execution of the fields
2282            **      was started before the parse routines found the
2283            **      closing parentheses), then a uOPNWM token (open
2284            **      parentheses loop with multiplier) is found in the
2285            **      reference counter field.  If such is the case, the
2286            **      uOPNWM token is replaced with a uOPNM- token to
2287            **      indicate that the loop counter has been decremented.
2288            **
2289            ** Algorithm:
2290            **      Move D1-4 to reference counter.
2291            **      If uOPNWM token in reference counter field, re-write
2292            **        with uOPNM-.
2293            **      Move D1-4 to loop counter.
2294            **      Decrement loop counter (DEC mode), replace; return.
2295            **
2296            ** History:
2297            **
2298            **    Date     Programmer             Modification
2299            **  --------   ----------   ----------------------------------
2300            **  12/08/82   MB           Documentation
2301            **
2302            ****************************************************************
```

```
2303            ****************************************************************
2304 1C177 1C3 =DCRMNT D1=D1- 4              To start of multiplier reserve.
2305 1C17A 310E        LC(2)  uOPNWM         To test if open loop.
2306 1C17E 14B         A=DAT1 B              Read reserve.
2307 1C181 966         ?A#C   B              Open loop?
2308 1C184 70          GOYES  DCRM03         No. Closing paren already found.
2309 1C186 CC          A=A-1  A              Yes. A(B)= uOPNM- .
2310 1C188 149         DAT1=A B              Re-write token; "cntr decremntd."
2311 1C18B 1C3  DCRM03 D1=D1- 4              To multiplier counter.
2312 1C18E D2   DCRM05 C=0    A              Zero nib 4.
2313 1C190 15F3        C=DAT1 4              Read counter.
2314 1C194 05          SETDEC                Counter is in decimal.
2315 1C196 CE          C=C-1  A              Decrement.
2316 1C198 04          SETHEX
2317 1C19A 15D3        DAT1=C 4              Re-write counter.
2318 1C19E 500         RTNNC
2319            *
2320 1C1A1 173         D1=D1+ 4              CRY= counter expired; renew.
2321 1C1A4 147         C=DAT1 A              Fetch reserve multiplier.
2322 1C1A7 1C3         D1=D1- 4              Replace expired counter.
2323 1C1AA 15D3        DAT1=C 4
2324 1C1AE DB          C=D    A              C= address to resume xqt.
2325 1C1B0 135         D1=C
2326 1C1B3 03          RTNCC                 Skip mult symbols.
2327            *
```

```
2328                    EJECT
2329              ********************************************************
2330              *
2331 1C1B5 7388   USGch/ GOSUB  =D12ROA        Save D1=xqt address in R0(A).
2332 1C1B9 8E00          GOSUBL =SENDEL
          00
2333 1C1BF 69D9   USG/1  GOTO   IMxq25
2334              *
2335              ********************************************************
2336              *
2337 1C1C3 7EAE   AVEDT1 GOSUB  StAVE+         A=curr xqt addr, D1=AvMemEnd.
2338 1C1C7 1CF    AVED.3 D1=D1- 16             To start of numeric expr.
2339 1C1CA 1577          C=DAT1 W              Read entire expr.
2340 1C1CE 03            RTNCC
2341              *
```

```
2342                    EJECT
2343              ************************************************************
2344              *
2345 1C1D0        CHKSKP
2346              *   NaN, Inf or -Inf. Skip all numeric symbols
2347              *   except for 3 (or 4 for -Inf) to output "NaN or "Inf".
2348              *
2349 1C1D0 109            R1=C                  D1=xqt address to R1.
2350 1C1D3 AF1            B=0    W              B(S)=flag, B(A)=counter for skips.
2351 1C1D6 96C            ?A#0   B              NaN?
2352 1C1D9 A0           GOYES  CHKSK1           Yes. 0=NaN.
2353              *
2354 1C1DB 948          ?A=0   S              No. -Inf?
2355 1C1DE 50           GOYES  CHKSK1           No. +Inf.
2356 1C1E0 A4D          B=B-1  S              B(S)=F: -Inf.
2357              *
2358 1C1E3 8E4E  CHKSK1 GOSUBL =CLOST         Clear 0 nib of status bits.
           7F
2359              *
2360 1C1E9 20    CHKSK3 P=     0
2361 1C1EB 1BD9         DO=(5) =IMtbl         Image character table.
           3C1
2362 1C1F2 7A3F         GOSUB  TstEn5         Does: D1=D1-2, A=DAT1 B.
2363              *---   D1=D1- 2              To next token.
2364              *---   A=DAT1 B              Read token.
2365 1C1F6 161   CHKSK5 DO=DO+ 2              To next table token (skip X!).
2366 1C1F9 14E          C=DAT0 B              Read table token.
2367 1C1FC 96A          ?C=0   B              End of table?
2368 1C1FF 36           GOYES  CHKSK7         Yes.
2369 1C201 966          ?A#C   B              Nope. Token match?
2370 1C204 2F           GOYES  CHKSK5         No. Try next one.
2371              *
2372 1C206       N/ISKP                       Yes! Count skips.
2373 1C206 876          ?ST=1  sSpecl         Skipping positions?
2374 1C209 11           GOYES  NISKP5         Yes.
2375 1C20B 7731         GOSUB  COUNTC         Increment skip count.
2376 1C20F 04           SETHEX
2377 1C211 57D          GONC   CHKSK3         (BET) Keep cntg Nan/Inf skips.
2378              *
2379 1C214 2C    NISKP2 P=     12             Display "*" to show overflow.
2380 1C216 64D9  NISKP3 GOTO   USGch4         Display blank.
2381              *
2382 1C21A 871   NISKP5 ?ST=1  sOvfFl         Overflow filling?
2383 1C21D 7F           GOYES  NISKP2         Yes.
2384 1C21F 870          ?ST=1  sStall         Stalling (no output)?
2385 1C222 7C           GOYES  CHKSK3         Yes.
2386 1C224 7701         GOSUB  CNTOUT         Count #skips for output.
2387 1C228 5DE          GONC   NISKP3         More skips. Display blank.
2388 1C22B 749F         GOSUB  AVEDT1         D1 to A, D1=AvMemEnd, C=expr.
2389 1C22F 96E          ?C#0   B              NaN?
2390 1C232 B0           GOYES  NISKP9         Yes.  P remains=0.
2391 1C234 28           P=     8              No. Inf or -Inf.
2392 1C236 94A          ?C=0   S              -Inf?
2393 1C239 40           GOYES  NISKP9         No.
2394 1C23B 2A           P=     10             Yes.
```

```
2395 1C23D 3DE4  NISKP9 LCASC  \fnI-NaN\      Pointer determines.
           16E4
           D294
           E666
2396 1C24D 15D7         DAT1=C 8              3 or 4 chars into Dat1.
2397 1C251 772E         GOSUB  ROu2CA         R0(9-5)=#chars to C(A). Sets P-0.
2398 1C255 D5           B=C    A              #chars to B.
2399 1C257 850          ST=1   sStall         "Stalling: no output."
2400 1C25A 6979         GOTO   USst05         Display "NaN" or "Inf".
2401               *
2402 1C25E 6398  IMx11j GOTO   IMxq11
2403               *
2404 1C262 876   CHKSK7 ?ST=1  sSpecl         Counting skips?
2405 1C265 9F           GOYES  IMx11j         No, outputting; xqt curr token.
2406 1C267 310D         LC(2)  uSTRPT         Yes.
2407 1C26B 962          ?A=C   B              String pointer?
2408 1C26E 13           GOYES  CHKS21         Yes. Skip 12 nibs over hex ptrs.
2409 1C270 E6           C=C+1  A              C(B)=uMULT.
2410 1C272 962          ?A=C   B              Multiplier?
2411 1C275 72           GOYES  CHKS19         Yes. Skip over 8 decimal nibbles.
2412 1C277 316E         LC(2)  EndNum         Token >= EndNum ends num fld.
2413 1C27B 9E2          ?A<C   B              Still within numeric fld?
2414 1C27E 42           GOYES  CHKS3j         Yes. Keep counting skips.
2415 1C280 D2           C=0    A
2416 1C282 303          LCHEX  3              C(A)=3= #chars in Nan or Inf.
2417               *
2418 1C285 949          ?B=0   S              -Inf?
2419 1C288 40           GOYES  CHKS15         No.
2420               *
2421 1C28A E6           C=C+1  A              Yes. Takes 4 chars.
2422 1C28C 05    CHKS15 SETDEC
2423 1C28E E1           B=B-C  A              Enough posns available?
2424 1C290 04           SETHEX
2425 1C292 431          GOC    USGOV+         No. "USING NaN Ovf" or "Inf Ovf".
2426 1C295 79E0         GOSUB  STORB          Yes. #chars to R0(9-5), B to R3.
2427 1C299 856          ST=1   sSpecl         "Outputting skips."
2428 1C29C 173   CHKS19 D1=D1+ 4              Past #digit storage.
2429 1C29F 1CB   CHKS21 D1=D1- 12
2430 1C2A2 664F  CHKS3j GOTO   CHKSK3         Start executing tokens.
2431               *
```

```
2432                    EJECT
2433                *****************************************************
2434                *
2435 1C2A6 119     USGOV+ C=R1
2436 1C2A9 135            D1=C                 Restore xqt address.
2437 1C2AC         USGOVF
2438 1C2AC D2              C=0    A             (To clear S7 for HTRAP.)
2439 1C2AE 0B              CSTEX                Save status bits in R1.
2440 1C2B0 109             R1=C
2441 1C2B3 22              P=     2             "Test OVF".
2442 1C2B5 8F00            GOSBVL =HTRAP        OVF set?
           000
2443                *-------------------------------------------
2444                * Return from HTRAP:
2445                *   If B(S)#0 then "Warning"      (OVF not set, no INX)
2446                *   Else [B(S)=0]
2447                *     If S7=1 then "Warning"      (OVF not set, but INX)
2448                *     Else [S7=0] then "Error"    (OVF set)
2449                *-------------------------------------------
2450 1C2BC 04              SETHEX
2451 1C2BE 137             CD1EX                Save D1 in R1;
2452 1C2C1 129             CR1EX                  status bits back to C.
2453 1C2C4 0B              CSTEX                Restore status bits; S7 to C.
2454 1C2C6 94D             ?B#0   S             Warning?
2455 1C2C9 50              GOYES  USGOV3        Yes.
2456 1C2CB A66             C=C+C  B             Maybe not. S7=1 from HTRAP?
2457                *
2458 1C2CE 20      USGOV3  P=     0
2459 1C2D0 3100            LC(2)  =eIMGOV       "IMAGE Ovfl".
2460 1C2D4 5C1             GONC   USGOV5        S7=0 from HTRAP.
2461 1C2D7 8F00            GOSBVL =MFWRQ8       Display warning.
           000
2462 1C2DE 8E00            GOSUBL =CKINFO       Reset output device if necessary.
           00
2463 1C2E4 856             ST=1   sSpecl        Spec'l digit output,
2464 1C2E7 851             ST=1   sOvfFl          display "*" for overflow.
2465 1C2EA 119             C=R1                 D1=xqt address back to C.
2466 1C2ED 6EA8            GOTO   IMxq27        Continue execution.
2467                *
2468 1C2F1 8CBF    USGOV5  GOLONG =kMFERR       Display error.
           1F
2469                *
```

```
2470                    EJECT
2471                ****************************************************************
2472                ****************************************************************
2473                **
2474                ** Name:(S) NXTEXP  -  Store pointers, execute next expression
2475                **
2476                ** Category:   EXCUTL
2477                **
2478                ** Purpose:
2479                **      Store pointer and status bits, call EXPEXC for IMAGE
2480                **      output items.
2481                **
2482                ** Entry:
2483                **      P      =  0
2484                **      D0=Program Counter (points to expression to be
2485                **        executed)
2486                **      D1=address of current BldIMG token
2487                **      RAM storage as shown in IMGxqt header
2488                **
2489                ** Exit:
2490                **      Through EXPEXC:
2491                **        D0=new Program Counter
2492                **        D1=points to item on math stack
2493                **
2494                ** Calls:      SetAVM, DT1C-A, EXPEXC
2495                **
2496                ** Uses:       EXPEXC can use anything
2497                **
2498                ** Stk lvls:   EXPEXC can use all levels (5 availble at call)
2499                **
2500                ** Algorithm:
2501                **      Save status bits in RAM at AvMemEnd+5.
2502                **      Save offset to D1 (current IMAGE token address) in
2503                **        RAM at AvMemEnd.
2504                **      Jump to EXPEXC.
2505                **
2506                ** History:
2507                **
2508                **    Date     Programmer              Modification
2509                **  --------   ----------   ----------------------------------
2510                **  12/08/82   MB           Documentation
2511                **
2512                ****************************************************************
2513                ****************************************************************
2514                *
2515 1C2F7 133  =NXTEXP AD1EX                  A(A)= xqt address.
2516 1C2FA 8EAF        GOSUBL =SetAVM
           6F
2517 1C300 174         D1=D1+ 5
2518 1C303 09          C=ST
2519 1C305 1553        DAT1=C X
2520 1C309 D6          C=A    A              C= xqt address.
2521 1C30B 790D        GOSUB  DT1C-A         Store offset in BldIMG.
2522 1C30F 8C00 =ExpExc GOLONG =EXPEXj       Expression Execute.
           00
```

```
2523                *
```

```
2524                    EJECT
2525              ********************************************
2526              *
2527 1C315        ExpEXP                          Expand exponent to 5 digit form.
2528 1C315 D4              A=B     A              Exponent to A.
2529 1C317 AD0             A=0     M              Clear out A(4-3).
2530 1C31A A34             A=A+A   X              Negative?
2531 1C31D 5A0             GONC    ExpEX1         No, positive. CRY cleared.
2532 1C320 938             ?A=0    X              Maybe. Exponent= 500?
2533 1C323 50              GOYES   ExpEX1         Yes. It's positive.
2534 1C325 A5C             A=A-1   M              Neg. A(M)= 999999999999.
2535              *
2536 1C328 AB4    ExpEX1   A=B     X              (Leave A= true exponent)
2537 1C32B E4              A=A+1   A              Return with A= exponent +1.
2538 1C32D 03              RTNCC
2539              *
```

```
2540                    EJECT
2541              ********************************************************
2542              *
2543 1C32F 11B    CNTOUT C=R3                    Counter to C(A).
2544 1C332 05            SETDEC
2545 1C334 CE            C=C-1  A               Count output posns.
2546 1C336 04            SETHEX
2547 1C338 580           GONC   CNTOT3          More output.
2548 1C33B 8E00          GOSUBL =csrw5          End of output. Restore R3.
           00
2549              *                             (CSRW5 preserves carry!)
2550 1C341 10B    CNTOT3 R3=C
2551 1C344 01            RTN                    Preserve carry!
2552              *
```

```
2553                    EJECT
2554            ********************************************************************
2555            ********************************************************************
2556            **
2557            ** Name:(S) COUNTC  -  Count output characters in IMAGE field
2558            **
2559            ** Category:   EXCUTL
2560            **
2561            ** Purpose:
2562            **      To count the number of output symbols in an IMAGE
2563            **      field.  Operates on individual symbols, checking
2564            **      to see if accompanied by a multiplier.  If not,
2565            **      increments count by 1; if so, adds multiplier
2566            **      value to count.
2567            **
2568            ** Entry:
2569            **      P      =  0
2570            **      D1 points to symbol which needs to be counted.
2571            **      B(A)=current count of symbols.
2572            **
2573            ** Exit:
2574            **      DEC mode!
2575            **      Carry clear
2576            **      If no multiplier accompanied symbol:
2577            **        P=0
2578            **        D1=same as entry (address+2 of next token to
2579            **          execute)
2580            **        B(S) incremented by 1
2581            **      If multiplier accompanied symbol:
2582            **        P=14
2583            **        D1 points to uLOOPB token (address+2 of next
2584            **          token to execute)
2585            **        B(A) incremented by multiplier value
2586            **
2587            ** Calls:      TstEn5
2588            **
2589            ** Uses.......
2590            **  Exclusive: A(B),B(A),C(A),D(A),P,D1
2591            **  Inclusive: same
2592            **
2593            ** Stk lvls:   1
2594            **
2595            ** NOTE:
2596            **      An application which processes the uMULT token by
2597            **      decrementing the loop counter will want to call
2598            **      the COUNTC subroutine as follows.  The HPIL ROM,
2599            **      for ENTER USING, is an example of an application
2600            **      which needs to call COUNTC this way.
2601            **         GOSBVL =COUNTC       Count symbol.
2602            **         SETHEX
2603            **         ?P=    0             Multiplier?
2604            **         GOYES  ..<exit>..    No.
2605            **         P=     0             Yes. Reset P.
2606            **         D1=D1+ 4             Fetch reference counter,
2607            **         DAT1=C 4               copy it into loop
```

```
2608            **        D1=D1- 4              counter.
2609            **        ..<exit>..
2610            **
2611            ** Algorithm:
2612            **      Move D1-2, to possible uLOOPB token.
2613            **      Test token for uLOOPB; if no match, reset D1+2, goto 2)
2614            **      (uLOOPB token found -- accompanying multiplier):
2615            **      Move D1+6 to reference counter.
2616            **      Read multiplier value into C(A).
2617            **      Reset D1 to uLOOPB token.
2618            **      Set P=14 to nullify LCHEX 1
2619            **  2)  LCHEX 1 for incrementing count
2620            **      Add B=B+C A for new count, in DEC mode
2621            **      Return, carry clear.
2622            **
2623            ** History:
2624            **
2625            **    Date     Programmer             Modification
2626            **  --------   ----------   ----------------------------------
2627            **  12/08/82   MB           Documentation
2628            **
2629            ********************************************************************
2630            ********************************************************************
2631            *
2632 1C346 D3   =COUNTC D=0    A             Force increment of
2633 1C348 CF          D=D-1  A               counter (see CNTP3).
2634 1C34A D2    COUNTP C=0    A
2635 1C34C 312D        LC(2)  uLOOPB
2636 1C350 7CDD        GOSUB  TstEn5         Does: D1=D1-2, A=DAT1 B.
2637            *---    D1=D1- 2             Check if multiplier.
2638            *---    A=DAT1 B             Read next token down.
2639 1C354 171         D1=D1+ 2             Restore D1 to "D" token.
2640 1C357 966         ?A#C   B             Multiplier?
2641 1C35A E0          GOYES  CNTP3         No. Only 1 D.
2642 1C35C 175         D1=D1+ 6             Yes. To mult reserve.
2643 1C35F 15F3        C=DAT1 4             Read multiplier.
2644 1C363 1C7         D1=D1- 8             D1 to uLOOPB.
2645 1C366 2E          P=     14            Nullify next LCHEX.
2646            *
2647 1C368 3110  CNTP3  LCHEX  01            (From above, now C(A)= 1.)
2648 1C36C 8BB         ?C<=D  A             How many more zeros?
2649 1C36F 60          GOYES  CNTP7         Still more.
2650 1C371 DB          C=D    A             Just that many.
2651 1C373 D3          D=0    A             To force CRY below.
2652 1C375 05    CNTP7 SETDEC
2653 1C377 C1          B=B+C  A             Increment #floats.
2654 1C379 8AB         ?D=0   A             Any more zeros?
2655 1C37C 00          RTNYES               No. End of float check. CRY set.
2656 1C37E E3          D=D-C  A             New #zeros.
2657 1C380 03          RTNCC                More zeros. CRY clear.
2658            *
2659            ************************************************
2660            *
2661 1C382 111   STORB  A=R1                 D1 (xqt addr) to A.
2662 1C385 131         D1=A                 Restore xqt address.
```

```
2663 1C388 8E88          GOSUBL =CSL2R0        C(A) to R0(9-5).
           6F
2664 1C38E 11B           C=R3
2665 1C391 8F00  STORB1  GOSBVL =CSLW5         Shift R3 left 5 times.
           000
2666 1C398 D9            C=B    A              Put B(A) in R3.
2667 1C39A 56A           GONC   CNTOT3         (BET) Save all in R3.
2668             *
```

```
2669                    EJECT
2670              *
2671              *-------- NOTE ----------------------------------------------*
2672              *---  This table is used for IMAGE parsing. Only characters  *
2673              *---    D . S M Z E C * P R   are really needed for CHKSKP,   *
2674              *---  but X " ' A and Z1 have been added for parsing. There  *
2675              *---  must be at least 16 entries in the table (including    *
2676              *---  the 00 byte, if necessary).  The order shown must be   *
2677              *---  maintained!!!                                          *
2678              *-------------------------------------------------------------*
2679              *                              Parse:
2680 1C39D        =IMtbl
2681 1C39D 85           CON(2) \X\             X                        0
2682 1C39F 44           CON(2) \D\             D                        1 *
2683 1C3A1 14           CON(2) \A\             A                        2
2684 1C3A3 E2           CON(2) \.\             .                        3 *
2685 1C3A5 22           CON(2) \"\             "                        4
2686 1C3A7 72           CON(2) \'\             '                        5
2687 1C3A9 35           CON(2) \S\             S                        6 *
2688 1C3AB D4           CON(2) \M\             M                        7 *
2689 1C3AD A5           CON(2) \Z\             Z                        8 *
2690 1C3AF 54           CON(2) \E\             E                        9 *
2691 1C3B1 34           CON(2) \C\             C                       10 *
2692 1C3B3 A2           CON(2) \*\             *                       11 *
2693 1C3B5 A5           CON(2) \Z\             Z1: unit's digit Z.     12
2694 1C3B7 05           CON(2) \P\             P                       13 *
2695 1C3B9 25           CON(2) \R\             R                       14 *
2696 1C3BB B4           CON(2) \K\             K  +----------------+   15
2697 1C3BD 84           CON(2) \H\             H  | Used in search |   15
2698 1C3BF 24           CON(2) \B\             B  |   at Nxtfld.   |   15
2699 1C3C1 E5           CON(2) \^\             ^  +----------------+   15
2700 1C3C3 00           CON(2) 0                Other tokens fall through.
2701              *
2702              *           -----  * See note above
2703              *
2704 1C3C5                 END
```

```
 AVE=D1  Ext                  -   713  2021
 AVED.3  Abs  115143 #1C1C7   -  2338
 AVEDT1  Abs  115139 #1C1C3   -  2337  1678  2388
 BYTscA  Abs  113226 #1BA4A   -    12   812  1496
 BldIM+  Abs  113258 #1BA6A   -    12   711  1730
=C+A2D1  Abs  114771 #1C053   -  1913  1199  1353  2149  2242
 CA2D1+  Abs  114774 #1C056   -  1914  1791
 CA2D1.  Abs  114777 #1C059   -  1915  1206
 CHKF11  Abs  114351 #1BEAF   -  1543  1502
 CHKFL5  Abs  114261 #1BE55   -  1493  1491  1558
 CHKFL7  Abs  114266 #1BE5A   -  1496  1535  1544  1570
 CHKFL9  Abs  114336 #1BEA0   -  1536  1517  1523  1529  1564
 CHKFLA  Abs  114361 #1BEB9   -  1552  1604
 CHKFLT  Abs  114251 #1BE4B   -  1489
 CHKS15  Abs  115340 #1C28C   -  2422  2419
 CHKS19  Abs  115356 #1C29C   -  2428  2411
 CHKS21  Abs  115359 #1C29F   -  2429  2408
 CHKS3j  Abs  115362 #1C2A2   -  2430  2414
 CHKSK1  Abs  115171 #1C1E3   -  2358  2352  2355
 CHKSK3  Abs  115177 #1C1E9   -  2360   809  2377  2385  2430
 CHKSK5  Abs  115190 #1C1F6   -  2365  2370
 CHKSK7  Abs  115298 #1C262   -  2404  2368
 CHKSKP  Abs  115152 #1C1D0   -  2345  1324
 CK"ON"  Ext                  -  2234
 CKINFO  Ext                  -  2015  2462
 CLOST   Abs  113101 #1B9CD   -    12  2358
 CLOST+  Abs  113098 #1B9CA   -    12   708  1904
 CNTOT3  Abs  115521 #1C341   -  2550  2547  2667
 CNTOUT  Abs  115503 #1C32F   -  2543  1588  1601  2386
 CNTP3   Abs  115560 #1C368   -  2647  2641
 CNTP7   Abs  115573 #1C375   -  2652  2649
=COUNTC  Abs  115526 #1C346   -  2632  2375
 COUNTP  Abs  115530 #1C34A   -  2634  1563
 CSL2RO  Abs  113174 #1BA16   -    12  1635  1929  2663
 CSLW5   Ext                  -  2665
 CSLW9j  Abs  113158 #1BA06   -    12   709
=CSRW9j  Abs  114815 #1C07F   -  1936   705  1627
 CSRWP9  Ext                  -  1936
 D12ROA  Abs  113212 #1BA3C   -    12  2331
 DCRMO3  Abs  115083 #1C18B   -  2311  2308
 DCRMO5  Abs  115086 #1C18E   -  2312  2240  2246
=DCRMNT  Abs  115063 #1C177   -  2304   938
 DECP=C  Ext                  -  1415
 DT1C-A  Abs  114712 #1C018   -  1726   706  2521
 DT1C.A  Abs  114717 #1C01D   -  1728   702
 ENDFL7  Abs  114429 #1BEFD   -  1588  1592
 ENDFLT  Abs  114419 #1BEF3   -  1584  1539  1541  1579
=ENDIMG  Abs  114752 #1C040   -  1902  2064
 ENDSTM  Abs  115000 #1C138   -  2156  2063
 ENDSTj  Abs  114932 #1C0F4   -  2063  2013
 EOLXCK  Ext                  -  2145
 EXPEXj  Ext                  -  2522
 EndNum  Abs     230 #000E6   -    12  1533  2412
 ExpEX1  Abs  115496 #1C328   -  2536  2531  2533
 ExpEXP  Abs  115477 #1C315   -  2527  1388  1435
```

```
=ExpExc  Abs  115471 #1C30F -  2522
 FLOATc  Abs  114391 #1BED7 -  1567  1520  1526
 FLOATd  Abs  114379 #1BECB -  1562  1499
 FLOTd5  Abs  114397 #1BEDD -  1570  1577  1581
 FLTDH   Ext                -  1700
 FLTstr  Abs  114401 #1BEE1 -  1575  1514
 FLTxsm  Abs  114404 #1BEE4 -  1576  1505  1508  1511
 FPOLLj  Abs  113047 #1B997 -    12   715
 FPOLLx  Abs  113043 #1B993 -    12   917  1331
 GETST1  Abs  114791 #1C067 -  1923  2022
 GETSTA  Abs  114787 #1C063 -  1922  1196  1903
=GetEXP  Abs  114822 #1C086 -  2012  1139  1347
 GtEXO1  Abs  114875 #1C0BB -  2038  2051
 GtEXO3  Abs  114880 #1C0C0 -  2040  2053
 GtEXO4  Abs  114886 #1C0C6 -  2041  2055
 GtEXO5  Abs  114891 #1C0CB -  2044  2037
 GtEXO9  Abs  114917 #1C0E5 -  2054  2046
 HTRAP   Ext                -  2442
 IMDO--  Abs  113188 #1BA24 -    12  1207
=IMGxq1  Abs  113323 #1BAAB -   698
=IMGxqt  Abs  113316 #1BAA4 -   695
 IMXPOL  Abs  113551 #1BB8F -   916   893
=IMXRTN  Abs  113557 #1BB95 -   919
=IMtbl   Abs  115613 #1C39D -  2680  2361
 IMx10j  Abs  114357 #1BEB5 -  1547  1482  1589  1602
 IMx11j  Abs  115294 #1C25E -  2402  2405
=IMxq05  Abs  113372 #1BADC -   720
 IMxq10  Abs  113382 #1BAE6 -   805   899  1146  1547
 IMxq11  Abs  113394 #1BAF2 -   811  2402
 IMxq12  Abs  113397 #1BAF5 -   812   807
 IMxq19  Abs  113547 #1BB8B -   899   928   939   947
 IMxq25  Abs  113561 #1BB99 -   925  1025  2333
=IMxq27  Abs  113564 #1BB9C -   926  1251  2065  2466
 InhEOL  Abs       4 #00004 -    12  1116
 InvArg  Ext                -  1704
 IvUSGj  Abs  111848 #1B4E8 -    12  2058
 MFWRQ8  Ext                -  2461
 N/ISKP  Abs  115206 #1C206 -  2372
 NANINF  Abs  113908 #1BCF4 -  1324  1358
 NISKP2  Abs  115220 #1C214 -  2379  2383
 NISKP3  Abs  115222 #1C216 -  2380  2387
 NISKP5  Abs  115226 #1C21A -  2382  2374
 NISKP9  Abs  115261 #1C23D -  2395  2390  2393
=NXTEXP  Abs  115447 #1C2F7 -  2515  2014
 NXTstm  Abs  115011 #1C143 -  2159  2157  2235
=NuOFF+  Abs  114726 #1C026 -  1783  1142  1706
 NuOFF3  Abs  114742 #1C036 -  1789  1784
=NuOFFS  Abs  114733 #1C02D -  1786  1250  1712
 Nxtfl3  Abs  112086 #1B5D6 -    12  1208
 PART3   Ext                -  2160
 POPMTH  Ext                -  2020
 POPTS+  Abs  114689 #1C001 -  1712  1215  1698
 POPTST  Abs  114693 #1C005 -  1713  2036
 PUTRES  Ext                -  2040
 Polrtn  Abs  113030 #1B986 -    12   919  1334
```

```
 ROu2CA  Abs  114812 #1C07C -  1935   720  1556  2397
 R3=D1+  Ext              -  1202
=RCVOFS  Abs  114768 #1C050 -  1911  1016
 REVPOP  Ext              -  1141  1228
 RND-12  Ext              -  1418
 SAVE++  Abs  114801 #1C071 -  1929  1464
 SENDEL  Ext              -  2158  2332
 SENDWD  Ext              -  1117
 SENDWj  Abs  113695 #1BC1F -  1116  1024  1249
 SET-ST  Abs  113103 #1B9CF -    12  1326
 SKPFLT  Abs  114453 #1BF15 -  1601  1615  1647
 STORB   Abs  115586 #1C382 -  2661  1585  2426
 STORB1  Abs  115601 #1C391 -  2665  1586
 STR$SB  Ext              -  1225
 SetAVE  Abs  113152 #1BA00 -    12  1783  1931  2147
 SetAVM  Abs  113146 #1B9FA -    12  1922  2516
 Setave  Abs  114808 #1C078 -  1931
 StAVE+  Abs  114805 #1C075 -  1930  1123  1660  1787  2337
 ToUSGn  Abs  113577 #1BBA9 -   933   898
 TstEn3  Abs  114949 #1C105 -  2133  2146
 TstEn5  Abs  114992 #1C130 -  2152  2362  2636
=TstEnd  Abs  114943 #1C0FF -  2131  2012  2062
 USG"*"  Abs  114483 #1BF33 -  1616   836  1609
 USG"A"  Abs  113704 #1BC28 -  1122   821
 USG"B"  Abs  114651 #1BFDB -  1698   881
 USG"C"  Abs  114537 #1BF69 -  1642   845
 USG"D"  Abs  114446 #1BF0E -  1597   815
 USG"E"  Abs  113632 #1BBE0 -  1080   860
 USG"H"  Abs  113808 #1BC90 -  1214   854
 USG"K"  Abs  113805 #1BC8D -  1213   857
 USG"M"  Abs  114571 #1BF8B -  1659   842
 USG"P"  Abs  114539 #1BF6B -  1643   848
 USG"S"  Abs  114569 #1BF89 -  1658   839  1623
 USG"Z"  Abs  114476 #1BF2C -  1612   818
 USG/1   Abs  115135 #1C1BF -  2333
 USGA01  Abs  113724 #1BC3C -  1130  1126
 USGB03  Abs  114678 #1BFF6 -  1706  1701  1703
 USGOV+  Abs  115366 #1C2A6 -  2435  2425
 USGOV3  Abs  115406 #1C2CE -  2458  2455
 USGOV5  Abs  115441 #1C2F1 -  2468  2460
 USGOVF  Abs  115372 #1C2AC -  2437  1447
 USGOVj  Abs  114160 #1BDF0 -  1446  1457  1461
 USGc0j  Abs  114533 #1BF65 -  1637  1652
 USGch*  Abs  113683 #1BC13 -  1108  1693
=USGch+  Abs  113685 #1BC15 -  1110  1133
=USGch-  Abs  113675 #1BC0B -  1105  1090
 USGch/  Abs  115125 #1C1B5 -  2331   884
 USGch0  Abs  113651 #1BBF3 -  1090  1637
 USGch1  Abs  113649 #1BBF1 -  1089
 USGch2  Abs  113647 #1BBEF -  1088
 USGch3  Abs  113645 #1BBED -  1087
 USGch4  Abs  113643 #1BBEB -  1086   830  1131  1591  1673  2380
 USGch5  Abs  113641 #1BBE9 -  1085   833  1654
 USGch6  Abs  113639 #1BBE7 -  1084   887
 USGch7  Abs  113637 #1BBE5 -  1083   890
```

```
 USGcp3  Abs  114565 #1BF85 -  1654  1645
 USGd03  Abs  114471 #1BF27 -  1608  1599
 USGdg1  Abs  114629 #1BFC5 -  1687  1684
 USGdg3  Abs  114639 #1BFCF -  1690  1688
 USGdg7  Abs  114644 #1BFD4 -  1692  1708
 USGdgt  Abs  114607 #1BFAF -  1678  1626
 USGdln  Abs  113756 #1BC5C -  1145   875
 USGend  Abs  114928 #1C0F0 -  2062   878
 USGlit  Abs  113735 #1BC47 -  1137   851
 USGlpb  Abs  113588 #1BBB4 -   943   827
 USGlpp  Abs  113592 #1BBB8 -   946   869
 USGlps  Abs  113590 #1BBB6 -   945   866
 USGmlt  Abs  113581 #1BBAD -   938   824
 USGnum  Abs  113912 #1BCF8 -  1326   933
=USGrst  Abs  113763 #1BC63 -  1195   872
 USGsm1  Abs  114588 #1BF9C -  1667  1664
 USGsm3  Abs  114590 #1BF9E -  1668  1666
 USGsm5  Abs  114603 #1BFAB -  1673  1670
 USGstr  Abs  113599 #1BBBF -  1015   863
 USGz*3  Abs  114496 #1BF40 -  1622  1632
 USGz*7  Abs  114501 #1BF45 -  1625  1620
 USer1j  Abs  114922 #1C0EA -  2058  1722  2039  2049
 USer1k  Abs  114708 #1C014 -  1722  1699  1906
 UShk05  Abs  113829 #1BCA5 -  1223  1216
 UShk09  Abs  113829 #1BCA5 -  1225  1219
 UShk15  Abs  113835 #1BCAB -  1227  1221
 UShk21  Abs  113867 #1BCCB -  1238  1243  1246
 UShk31  Abs  113893 #1BCE5 -  1248  1230  1239
 UShk35  Abs  113900 #1BCEC -  1250  1349
=USloop  Abs  115019 #1C14B -  2233   946
 USnm03  Abs  113931 #1BD0B -  1346  1330
=USnm05  Abs  113938 #1BD12 -  1348
 USnm07  Abs  113999 #1BD4F -  1370  1366  1368
 USnm11  Abs  114026 #1BD6A -  1381  1376
 USnm15  Abs  114031 #1BD6F -  1383  1372
 USnm17  Abs  114034 #1BD72 -  1385  1374  1379
 USnm25  Abs  114058 #1BD8A -  1406  1394  1396
 USnm33  Abs  114068 #1BD94 -  1410  1386  1407
 USnm39  Abs  114089 #1BDA9 -  1416  1414
 USnm41  Abs  114111 #1BDBF -  1424  1419
 USnm43  Abs  114147 #1BDE3 -  1441  1438
 USnm47  Abs  114167 #1BDF7 -  1449  1427  1445
 USnm51  Abs  114183 #1BE07 -  1456  1451
 USnm53  Abs  114193 #1BE11 -  1460  1454
 USnm55  Abs  114198 #1BE16 -  1462  1432  1453  1459
 USnm57  Abs  114219 #1BE2B -  1472  1470
 USnm59  Abs  114230 #1BE36 -  1476  1473
=USst03  Abs  113614 #1BBCE -  1020  1113
=USst05  Abs  113620 #1BBD4 -  1022  2400
 csrm5   Ext                -  2548
 eIMGOV  Ext                -  2459
 kMFERR  Abs  111854 #1B4EE -    12  2468
 pIMXCH  Ext                -   918
 pIMXQT  Ext                -   716
 pIMcpu  Ext                -  1333
```

```
s"S,M"  Abs      0 #00000 -    45  1373  1383  1490  1538  1578  1608  1622
                                1672
sC/P    Abs      1 #00001 -    12    46    56
sCntg   Abs      2 #00002 -    12    47
sCplxP  Abs      7 #00007 -    12    53
sCplxW  Abs      7 #00007 -    53  1329
sFLOAT  Abs      1 #00001 -    46  1446  1481  1598  1607  1614  1646  1663
sFOUND  Abs     10 #0000A -    12  1905
sInit   Abs      3 #00003 -    12    48
sKnotH  Abs      3 #00003 -    50  1213  1220  1229
sNUME   Abs      2 #00002 -    47  1080  1385  1426  1439  1472  1475  1536
                              1603  1606  1665  1683
sOvfFl  Abs      1 #00001 -    56  2382  2464
sRturn  Abs      5 #00005 -    51   805   927   930  1590
sSTOP   Abs      5 #00005 -    12    51
sSpecl  Abs      6 #00006 -    12   806  1346  2038  2041  2052  2373  2404
                              2427  2463
sStall  Abs      0 #00000 -    55  2384  2399
sXQT    Abs      0 #00000 -    12    45    55
sZeros  Abs      3 #00003 -    48    50  1138  1348  1489  1562  1576  1625
                              1628  1634  1644  2050  2054
tEOL    Ext              -  2134
uALit   Abs    247 #000F7 -    12   850
uDELIM  Abs    244 #000F4 -    12   874
uHKB^   Abs    246 #000F6 -    12   896
uIMXCH  Abs    212 #000D4 -    12   892
uIMend  Abs    240 #000F0 -    12   877
uLOOPB  Abs    210 #000D2 -    12   826  2635
uLOOPP  Abs    239 #000EF -    12   868
uLOOPS  Abs    211 #000D3 -    12   865
uMULT   Abs    209 #000D1 -    12   823  1501
uOPNWM  Abs    224 #000E0 -    12  2305
uRESTP  Abs    241 #000F1 -    12   696   871
uSTRPT  Abs    208 #000D0 -    12   862  1513  2406
```

Input Parameters

  Source file name is MB&USG::MS

  Listing file name is MB/USG:TI:ML::-1

  Object file name is MB%USG:TI:MS::-1

                                111111
                       0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                      TITLE  Graphics Control Routines
 2 1C3C5                ABS    #1C3C5
 3              *    SSS   BBBB    &      GGG   PPPP   H   H
 4              *   S   S  B   B  & &    G   G  P   P  H   H
 5              *   S      B   B  & &    G      P   P  H   H
 6              *    SSS   BBBB    &     G GGG  PPPP   HHHHH
 7              *       S  B   B  & & &  G   G  P      H   H
 8              *   S   S  B   B  &  &   G   G  P      H   H
 9              *    SSS   BBBB    && &   GGG   P      H   H
10
11                      RDSYMB SB%DSP::MS
12             ********************************************************************
13             ********************************************************************
14             **
15             ** Name:(S) GDISP$  -  GDISP$ function execution
16             **
17             ** Category:   FNEXEC
18             **
19             ** Purpose:
20             **      Implements GDISP$ function
21             **
22             ** Entry:
23             **      P      =  0
24             **      D0 is program counter
25             **      D1 is stack pointer
26             **
27             ** Exit:
28             **      Exits through EXPR
29             **
30             ** Calls:      CPYDD-
31             **
32             ** Algorithm:
33             **      Save D0 on stack
34             **      Calculate where stack item will start
35             **      If not enough memory then
36             **        Exit with "Insufficient Memory" error
37             **      Write out header for 132 character string
38             **      Copy rightmost display driver (DD) to string
39             **      Copy middle DD to string
40             **      Copy leftmost DD to string
41             **      Point stack pointer to new string
42             **      Restore D0 from stack
43             **      Exit through EXPR
44             **
45             ** History:
46             **
47             **    Date     Programmer             Modification
48             **   --------   ----------   ----------------------------------
49             **   10/26/83   B.S.         Added documentation
50             **
51             ********************************************************************
52             ********************************************************************
53 1C3C5 00            NIBHEX 00              Argument Count=[0,0]
54 1C3C7 136  =GDISP$ CDOEX
55 1C3CA 06            RSTK=C                 Save PC
```

```
     56 1C3CC 1B00          DO=(5) =AVMEMS
              000
     57 1C3D3 146           C=DATO A              Read (AVMEMS)
     58 1C3D6 D5            B=C    A              Save it in B(A)
     59 1C3D8 133           AD1EX                 A(A)=Stack pointer
     60 1C3DB 3481          LC(5)  132*2+16       Length of stack item
              100
     61                                           (132 bytes + header)
     62 1C3E2 EA            A=A-C  A              Calculate end of memory required
     63 1C3E4 8B4           ?A<B   A              Is this much available?
     64 1C3E7 25            GOYES  MemErJ         No, then give memory error
     65 1C3E9 131           D1=A                  D1 points where string header
     66                                              goes
     67 1C3EC 100           R0=A                  Save pointer to string header
     68 1C3EF 36F0          LC(7)  (132*2)~#0F    String header value
              8010
              0
     69 1C3F8 1557          DAT1=C W              Write out string header
     70 1C3FC 17F           D1=D1+ 16             Skip past string header
     71 1C3FF 1B00          DO=(5) =DD1END
              000
     72 1C406 D1            B=0    A              B(B)=0 Start of this DD
     73 1C408 7330          GOSUB  CPYDD-         Copy display driver to stack
     74 1C40C 1B00          DO=(5) =DD2END
              000
     75 1C413 7820          GOSUB  CPYDD-         Copy display driver to stack
     76 1C417 304           LCHEX  4
     77 1C41A A85           B=C    P              B(B)=04 Start of this DD
     78 1C41D 1B00          DO=(5) =DD3END
              000
     79 1C424 7710          GOSUB  CPYDD-
     80 1C428 110           A=R0                  Recall new stack pointer
     81 1C42B 131           D1=A                  Put it back in D1
     82 1C42E 07            C=RSTK                Recall PC
     83 1C430 134           DO=C                  Restore PC to DO
     84 1C433 8C00          GOLONG =expr
              00
     85            *-
     86            *-
     87 1C439 8C00  MemErJ  GOLONG =MEMERj        GOTO MEMERR
              00
     88            *-
     89            *-
     90            * CPYDD-  copys a display driver into the stack
     91            *   Entry: B(B) = lower 2 digits of lowest valid address in DD
     92            *          DO points past highest address in this DD
     93            *          D1 points to last char of string for this DD
     94            *
     95 1C43F 181   CPYDD- DO=DO- 2              Move display driver pointer
     96 1C442 14E          C=DATO B              Read a byte from display driver
     97 1C445 14D          DAT1=C B              Write it to stack
     98 1C448 171          D1=D1+ 2              Move to next stack byte
     99 1C44B 136          CDOEX
    100 1C44E 134          DO=C                  Copy display driver pointer to C
    101 1C451 965          ?B#C   B              At end of display driver?
```

```
102 1C454 BE          GOYES  CPYDD-          No, then loop back
103 1C456 03          RTNCC                  Yes, then return
104             *-
105             ************************************************************
106             ************************************************************
107             **
108             ** Name:    GDISP   -  GDISP statement execution
109             **
110             ** Category:   STEXEC
111             **
112             ** Purpose:
113             **      Implements GDISP statement
114             **
115             ** Entry:
116             **      P      =  0
117             **      DO points past GDISP token
118             **
119             ** Exit:
120             **      Exits through NXTSTM
121             **
122             ** Calls:      EXPEXC,POP1S,CPY-DD,RCLSTA,STOSTA
123             **
124             ** Algorithm:
125             **      Evaluate string expression
126             **      Pop string from stack
127             **      Copy first part of string to leftmost DD
128             **      Copy second part of string to middle DD
129             **      Copy third part of string to rightmost DD
130             **      Set BitsOK display status so display won't be changed
131             **      Set NEEDSC to indicate that graphics are in display
132             **      Exit through NXTSTM
133             **
134             ** History:
135             **
136             **    Date     Programmer             Modification
137             **  --------   ----------   ----------------------------------
138             **  10/26/83   B.S.         Added documentation
139             **
140             ************************************************************
141             ************************************************************
142 1C458 0000         REL(5) =DELAYd
        0
143 1C45D 0000         REL(5) =STRNGP
        0
144 1C462 7000 =GDISP  GOSUB  =ExpExc        Evaluate string expression
145 1C466 8F00         GOSBVL =POP1S         Pop a string from stack
        000
146 1C46D 137          CD1EX
147 1C470 D7           D=C    A              Save end of string in D(A)
148 1C472 C2           C=C+A  A              Calculate start of string
149 1C474 135          D1=C                  Point D1 at start of string
150 1C477 1B00         DO=(5) =DD3ST         Point DO at start of leftmost DD
        000
151 1C47E 3100         LC(2)  =DD3END        End of this DD (and next one)
152 1C482 7A30         GOSUB  CPY-DD         Copy to this DD
```

```
153 1C486 1A00          D0=(4) =DD2ST          Point to start of next DD
          00
154 1C48C 7230          GOSUB  CPY-D1          Copy to this DD
155 1C490 1A00          D0=(4) =DD1ST          Point to start of rightmost DD
          00
156 1C496 3100          LC(2)  =DD1END         End of this DD
157 1C49A 7220          GOSUB  CPY-DD          Copy to this DD
158 1C49E 8F0A          GOSBVL =RCLSTA         Recall display status
          B10
159 1C4A5 851           ST=1   BitsOK          Pretend display is exactly right
160 1C4A8 8FBA          GOSBVL =STOSTA         Store back display status
          B10
161 1C4AF 1A00          D0=(4) =NEEDSC
          00
162 1C4B5 30E           LCHEX  E               Indicate graphics is in display
163 1C4B8 1540          DAT0=C P
164 1C4BC 6000          GOTO   =POKE25         GOTO  NXTSTM
165             *-
166             *-
167             * CPY-DD -  Copy to display driver, fills in zeros if past
168             *            end of string.
169             *   Entry: C(B) is low nibbles of highest valid address in DD
170             *          D0 points to lowest address of this DD
171             *          D1 points into stack at start of data for this DD
172             *          D(A) points to end of string
173             *   Exit:  B(B) is C(B) on entrance
174             *          D0 points past end of DD
175             *
176             *
177 1C4C0 D5    CPY-DD B=C    A              Move end of DD to B(B)
178 1C4C2 1C1   CPY-D1 D1=D1- 2              Move stack pointer
179 1C4C5 137          CD1EX                 Swap stack pointer to C(A)
180 1C4C8 8B3          ?C<D   A              Past end of stack item?
181 1C4CB B1           GOYES  CPY-D3         Yes, then fill in zero
182 1C4CD 137          CD1EX                 No, then swap back stack pointer
183 1C4D0 14B          A=DAT1 B              Read byte from stack
184 1C4D3 148   CPY-D2 DAT0=A B              Write byte to display driver
185 1C4D6 161          D0=D0+ 2              Move DD pointer
186 1C4D9 136          CD0EX
187 1C4DC 134          D0=C                  Copy DD pointer to C(A)
188 1C4DF 965          ?C#B   B              At end of this display driver?
189 1C4E2 0E           GOYES  CPY-D1         No, then loop back
190 1C4E4 03           RTNCC                 Yes, then return
191             *-
192             *-
193 1C4E6 137   CPY-D3 CD1EX                 Swap back stack pointer
194 1C4E9 D0           A=0    A              Fill in zero byte
195 1C4EB 47E          GOC    CPY-D2         (B.E.T.)
196 1C4EE              END
```

AVMEMS	Ext			-	56				
BitsOK	Abs	1	#00001	-	11	159			
CPY-D1	Abs	115906	#1C4C2	-	178	154	189		
CPY-D2	Abs	115923	#1C4D3	-	184	195			
CPY-D3	Abs	115942	#1C4E6	-	193	181			
CPY-DD	Abs	115904	#1C4C0	-	177	152	157		
CPYDD-	Abs	115775	#1C43F	-	95	73	75	79	102
DD1END	Ext			-	71	156			
DD1ST	Ext			-	155				
DD2END	Ext			-	74				
DD2ST	Ext			-	153				
DD3END	Ext			-	78	151			
DD3ST	Ext			-	150				
DELAYd	Ext			-	142				
ExpExc	Ext			-	144				
=GDISP	Abs	115810	#1C462	-	144				
=GDISP$	Abs	115655	#1C3C7	-	54				
MEMERj	Ext			-	87				
MemErJ	Abs	115769	#1C439	-	87	64			
NEEDSC	Ext			-	161				
POKE25	Ext			-	164				
POP1S	Ext			-	145				
RCLSTA	Abs	7072	#01BA0	-	11	158			
STOSTA	Abs	7083	#01BAB	-	11	160			
STRNGP	Ext			-	143				
expr	Ext			-	84				

Input Parameters

  Source file name is SB&GPH::MS

  Listing file name is SB/GPH:TI:ML::-1

  Object file name is SB%GPH:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                *     SSS    GGG    &     PPPP   00000  K   K
                *    S   S  G   G  & &    P   P  0   0  K  K
                *    S      G      & &    P   P  0   0  K K
                *     SSS   G GGG   &     PPPP   0   0  KK
                *        S  G   G  & & &  P      0   0  K K
                *    S   S  G   G  &  &   P      0   0  K  K
                *     SSS    GGG    && &  P      00000  K   K

                        TITLE  POKE, PEEK$, and ADDR$ <831216.1616>
 1C4EE                  ABS    #1C4EE
                        RDSYMB SB%RAM::MS
                        RDSYMB TI%EQU::MS
```

```
13                    STITLE POPNUM
14                 ****************************************************************
15                 ****************************************************************
16                 **
17                 ** Name:    POPNUM  -  Pop Positive Real and Convert to Hex
18                 **
19                 ** Category:   CONVRT
20                 **
21                 ** Purpose:    Pops number off math stack, ensures it's real,
22                 **             and converts it to hex.
23                 **
24                 ** Entry:      Data on stack pointed to by D1
25                 **
26                 ** Exit:
27                 **             A contains hex equivalent
28                 **             Number popped off stack (D1 points past it)
29                 **             Carry set
30                 **             P=0
31                 **             F-R0-0 contains D0 at entry
32                 **               ERROR exits:
33                 **                 eDATTY - if complex
34                 **                 eIVARG - if number is negative or too large
35                 **
36                 ** Calls:      POP1R, FLTDH, SAVD0
37                 **
38                 ** Uses.......
39                 **  Inclusive: A-C, D1,D0, XM, F-R0-0
40                 **
41                 ** Stk lvls:   2
42                 **
43                 ** History:
44                 **
45                 **    Date     Programmer              Modification
46                 **  --------   ----------   ----------------------------------
47                 **  07/19/82   S.W.         Wrote routine
48                 **  10/19/82   S.W.         Added call to POP1R, SAVD0
49                 **
50                 ****************************************************************
51                 ****************************************************************
52                 *
53 1C4EE 8F00  POPNUM GOSBVL =POP1R
          000
54 1C4F5 17F          D1=D1+ 16
55 1C4F8 7B80         GOSUB  SAVD0
56 1C4FC 8E00         GOSUBL =FLTDH         CONVERT TO HEX
          00
57 1C502 400          RTNC                  If positive & not too big
58                 *
```

```
                        STITLE PEEK$ Function Execution
                    ****************************************************************
                    ****************************************************************
                    **
                    ** Name:     PEEK$   -  PEEK$ Function Execution
                    **
                    ** Category:   FNEXEC
                    **
                    ** Purpose:    Executes PEEK$ Function
                    **
                    ** Entry:      Address & #nibbles to PEEK are on stack
                    **
                    ** Exit:       String pushed on stack
                    **             Exit via PCEXPR
                    **                Error Exits:
                    **                  eDATTY - complex numeric expr
                    **                  eARGOR - negative numeric expr
                    **                           too large
                    **                  eIVARG - string expr not a proper address
                    **                  ePROT  - PRIVACY violation
                    **                  eMEM   - Insufficient memory
                    **
                    ** Calls:    ADDRCK, FLTDH, POPNUM, ADRPRT(R0,R2), REV$,
                    **           HEXASC, STRHDR, SAVD1, RSTD1
                    **
                    ** Uses:     A-D, S8, R0, R1, R2 (ADRPRT), F-R0-0, F-R0-1
                    **
                    ** Detail:   NUMBER OF NIBBLES REQUESTED CAN'T EXCEED 7FFFF.
                    **           PEEK$ INTO NON-EXISTENT MEMORY RETURNS ZEROES.
                    **           PEEK$ INTO PRIVATE FILE SPACE ERRORS.
                    **
                    **        PEEK$(address,#nibs)
                    **             where <address> is a string expr
                    **                   <#nibs> is a numeric expr interpreted
                    **                           as number of nibbles
                    **                                  (in decimal)
                    ** Stack lvls: 4  (ADDRCK uses 3)
                    **
                    ** Note:       PEEK$ does a wrap-around if data pointer
                    **             carries (FFFFF -> 0)
                    **
                    ** History:
                    **
                    **    Date     Programmer   Modifications
                    **  --------   ----------   -------------
                    **  07/04/82   S.W.         Added documentation
                    **
                    ****************************************************************
                    ****************************************************************
                    *
1C505 6311  PK$E24  GOTO   INVARG
1C509 8422          NIBHEX 8422
                    *
1C50D 7DDF =PEEK$   GOSUB  POPNUM
1C511 D6            C=A    A
```

```
114 1C513 C6          C=C+C  A            ARG OUT OF RANGE?
115 1C515 4FE         GOC    PK$E24       (>7FFFF)
116 1C518 100         R0=A                SAVE #NIBS IN R0
117 1C51B 7680        GOSUB  ADDRCK       POP ADDRESS OFF STACK
118 1C51F 7550        GOSUB  SAVD1        SAVE D1 in F-R0-1
119           * ADDRESS IN B(A)
120 1C523 848         ST=0   8            WANT TO READ ONLY
121 1C526 110         A=R0                #NIBS REQUESTED
122 1C529 7271        GOSUB  ADRPRT
123           *
124           * OKAY TO DO READ (MEMORY PERMITTING) - PUSH DATA ON STACK
125           * BLOCK START TO READ IN R0
126           *
127 1C52D DA          A=C    A            A=#NIBS
128           * A=C=D=#nibs to read
129 1C52F C4          A=A+A  A            #NIBS TO PUSH ON STACK
130 1C531 7160        GOSUB  RSTD1        RESTORE D1
131 1C535 D6          C=A    A
132 1C537 8F00        GOSBVL =STRHDR
          000
133 1C53E 110         A=R0
134 1C541 130         D0=A                Start of block to read
135           * MUST CONVERT EACH NIBBLE IN RAM TO CORRESPONDING ASCII
136 1C544 CF    PK$20 D=D-1  A
137 1C546 4C1         GOC    PK$60        Done?
138 1C549 AC2         C=0    S            1 nib at a time
139 1C54C 15A0        A=DAT0 1
140 1C550 8E00        GOSUBL =HEXASC
          00
141 1C556 149         DAT1=A B
142 1C559 171         D1=D1+ 2
143 1C55C 160         D0=D0+ 1
144 1C55F 64EF        GOTO   PK$20        If D0 carries,wrap around
145           * STRING & HEADER WRITTEN OUT TO STACK
146           * PT. D1 TO HEADER
147 1C563 119   PK$60  C=R1
148 1C566 135         D1=C
149 1C569 20    PK$70 P=     0
150 1C56B 8E00        GOSUBL =REV$
          00
151 1C571 8D00        GOVLNG =PCEXPR      Restore PC from F-R0-0
          000
152           *                            & GOVLNG to EXPR
153 1C578 137 =SAVD1  CD1EX
154 1C57B 1F0A        D1=(5) =F-R0-1
          8F2
155 1C582 145         DAT1=C A
156 1C585 01          RTN
157           *
158 1C587 136 =SAVD0  CD0EX
159 1C58A 1BB9        D0=(5) =F-R0-0
          8F2
160 1C591 144         DAT0=C A
161 1C594 01          RTN
162           *
```

```
163 1C596 1F0A =RSTD1  D1=(5) =F-R0-1
          8F2
164 1C59D 147          C=DAT1 A
165 1C5A0 135          D1=C
166 1C5A3 01           RTN
167           *
```

```
168                     EJECT
169             ****************************************************************
170             ****************************************************************
171             **
172             ** Name:     ADDRCK  -  Address Check
173             **
174             ** Category:   LOCAL
175             **
176             ** Purpose:    Pops string expression off stack, checks its
177             **             validity as a hex address, and returns it
178             **             as a hex address
179             **
180             ** Entry:      D1 points to header of string expression on
181             **             the math stack
182             **
183             ** Exit:       Valid address in B(A)
184             **             D1 points past string
185             **             P=0
186             **                else
187             **                 ERROR EXIT.
188             **
189             ** Calls:      REVPOP, ASCHEX
190             **
191             ** Uses:       A, B, C(A), D(A), D1, D0
192             **
193             ** Stack lvls: 3
194             **
195             ** Detail:     REV$ uses  A, B, C(A), D(A), D1, D0
196             **             addresses can't be strings of over 5 characters.
197             **
198             ** History:
199             **
200             **    Date     Programmer   Modifications
201             **  --------   ----------   -------------
202             **  07/04/82   S.W.         Improved documentation.
203             **  07/05/82   S.W.         Modified code to eliminate call
204             **                           to POP1S - takes advantage of
205             **                           the fact REV$ calls POP1S.
206             **                           Also added null string check.
207             **  10/21/82   S.W.         Replaced call to REV$ w/ REVPOP
208             **
209             ****************************************************************
210             ****************************************************************
211             *
212             *
213 1C5A5 8F00  ADDRCK GOSBVL =REVPOP        REVERSE STRING ON STACK
        000
214 1C5AC D2           C=0    A
215 1C5AE 30A          LCHEX  A
216 1C5B1 8B6          ?A>C   A             MORE THAN 5 CHARACTERS?
217 1C5B4 56           GOYES  INVARG
218 1C5B6 D6           C=A    A             COPY STRING LENGTH INTO C(0)
219 1C5B8 80D0         P=C    0
220 1C5BC 0D           P=P-1
221 1C5BE 4A5          GOC    INVARG        NULL String?
```

```
222 1C5C1 133          AD1EX
223 1C5C4 C2           C=C+A  A              POSITION PAST STRING
224 1C5C6 131          D1=A                 RESTORE D1
225 1C5C9 AF0          A=0    W
226 1C5CC 1531         A=DAT1 WP
227 1C5D0 135          D1=C                 PT D1 PAST STRING
228 1C5D3 D1           B=0    A             WAS B=0 W (HISTORICALLY LESS CODE)
229 1C5D5 20           P=     0
230             * MAKE SURE ASCII CHARACTER A LEGAL HEX DIGIT & CONVERT IT
231 1C5D7 7800  ADRCK2 GOSUB  ASCHEX
232 1C5DB 968          ?A=0   B              DONE?
233 1C5DE 00           RTNYES
234 1C5E0 56F          GONC   ADRCK2         (B.E.T.)
235             *
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:    ASCHEX  -  Ascii to Hex Conversion
                **
                ** Category:   CONVRT
                **
                ** Purpose:    Converts an ascii character to a hex digit
                **
                ** Entry:      A(B) contains ascii character
                **             P=0
                **
                ** Exit:       B(0) = Hex Digit
                **             B(W) shifted left before reading in digit
                **             Ascii character shifted off A via 2 word shifts
                **             P=0
                **             Carry clear
                **                ERROR EXIT IFF:
                **                  eIVARG - A(B) not digit, or A-F, or a-f
                **
                ** Calls:      RANGE, CONVUC
                **
                ** Uses:       A, B, C
                **
                ** Stack lvls: 1
                **
                ** History:
                **
                **    Date     Programmer   Modifications
                **  --------   ----------   ------------
                **  07/04/82   S.W.         Added documentation
                **
                ********************************************************************
                ********************************************************************
                *
1C5E3 8E00      ASCHEX GOSUBL =DRANGE
      00
1C5E9 401               GOC    ASCHX5
                *
1C5EC BF1       ASCHX2 BSL    W
1C5EF A88              B=A    P
1C5F2 BF4              ASR    W
1C5F5 BF4              ASR    W
1C5F8 01               RTN
                *
1C5FA 8E00      ASCHX5 GOSUBL =CONVUC
      00
1C600 3314             LCASC  \FA\
      64
1C606 8E00             GOSUBL =RANGE
      00
1C60C 4C0              GOC    INVARG          NOT LETTER OR DIGIT?
1C60F 3173             LCHEX  37
1C613 B6A              A=A-C  B
```

```
287 1C616 55D          GONC   ASCHX2         (B.E.T.)
288               *
289 1C619 8C00  INVARG GOLONG =InvArg
          00
290               *
291               *
```

```
                        STITLE POKE Statement Execution
                ************************************************************
                ************************************************************
                **
                ** Name:    POKE    -  Execution of POKE statement
                **
                ** Category:   STEXEC
                **
                ** Purpose:    Executes POKE statement
                **
                ** Entry:      D0 points at string expr for POKE address
                **
                ** Exit:       via NXTSTM
                **              ERROR EXITS:
                **                  eIVARG - invalid address
                **                  eFPROT  - PRIVACY or SECURITY violation
                **                  eFACCS - illegal access
                **
                ** Calls:      EXPEXC, ADDRCK, POP1S, ADRPRT (R0,R2), ASCHEX
                **
                ** Uses:       A-D, R0-R4, S-R0-0, All of function scratch
                **
                ** Stack lvls: 5
                **
                ** Detail:     POKE into ROM or non-existent memory NOPS.
                **             POKE into SECURE or PRIVATE file errors.
                **             POKE into any file's chain offset or into 1st 8
                **             nibbles of any IRAM generates error.
                **
                ** Note:  POKE doesn't wrap around.  POKE terminates past FFFFF
                **
                ** History:
                **
                **    Date     Programmer   Modifications
                **  --------   ----------   -------------
                **  07/04/82   S.W.         Improved documentation
                **
                ************************************************************
                ************************************************************
                *
                *
1C61F 00000             REL(5) =FIXDC
      0
1C624 00000             REL(5) =POKEP
      0
1C629 7000  =POKE       GOSUB  =ExpExc          THROW STRING ON STACK
1C62D 747F              GOSUB  ADDRCK           CHECK VALIDITY OF ADDRESS
                * STARTING ADDRESS IN B(A) - SAVE IT IN SCRATCH RAM
1C631 1F17              D1=(5) =S-R0-0
      8F2
1C638 D4                A=B    A
1C63A 141               DAT1=A A
1C63D 161               D0=D0+ 2                STEP OVER COMMA TOKEN
1C640 8F00              GOSBVL =EXPEX-          THROW DATA ON STACK
      000
```

```
343                *                                (starting @ TFORN)
344                * MAKE SURE IT'S A STRING
345 1C647 8F00             GOSBVL =POP1S            MAKE SURE IT'S A STRING
          000
346 1C64E 25               P=     5
347 1C650 A80              A=0    P                 WANT 6TH NIB ZEROED
348 1C653 20               P=     0
349 1C655 81C              ASRB                     #NIBS IN AFFECTED BLOCK
350 1C658 1B17             D0=(5) =S-R0-0
          8F2
351 1C65F 146              C=DAT0 A                 ADDR TO WRITE TO
352 1C662 D5               B=C    A
353 1C664 858              ST=1   8                 INTEND TO WRITE TO ADDR. SPACE
354 1C667 7430             GOSUB  ADRPRT
355                * MUST CONVERT FROM ASCII TO DATA BEFORE WRITING OUT
356                * C=#NIBS
357                * WANT D0=END SOURCE OF DATA TO POKE
358                * WANT D1=BGN DESTINATION OF WHERE DATA TO GO
359 1C66B 131              D1=A                     BGN DEST.
360 1C66E 1BE9             D0=(5) =TFORN
          5F2
361 1C675 142              A=DAT0 A
362 1C678 130              D0=A                     END SOURCE
363 1C67B CF     POKE10    D=D-1  A
364 1C67D 4B1              GOC    POKE25
365 1C680 181              D0=D0- 2
366 1C683 AC2              C=0    S
367 1C686 14A              A=DAT0 B
368 1C689 765F             GOSUB  ASCHEX
369 1C68D D4               A=B    A
370 1C68F 1590             DAT1=A 1                 WRITE OUT 1 NIB
371 1C693 170              D1=D1+ 1
372 1C696 54E              GONC   POKE10            If carry, terminate POKE
373                *
374 1C699 8C00   =POKE25   GOLONG =REN180           GOTO NXTSTM
          00
375                *
```

```
                EJECT
*****************************************************************
*****************************************************************
**
** Name:    ADRPRT  -  Address Protect
**
** Category:   LOCAL
**
** Purpose:    Protects address space from PEEK$ and POKE
**
** Entry:      B(A) = Block Start
**             A(A) = Block size (in nibbles)
**             P=0
**             S8=1 => INTEND TO WRITE
**
** Exit:       Error exits if:
**              1) PROTECTION PROHIBITS OPERATION.
**                a)POKE on PRIVATE or SECURE file.
**                   Can't POKE header or data.
**                b)PEEK on PRIVATE file.
**                   Can't PEEK header or data.
**              2) ILLEGAL ACCESS (POKE) - File
**                 chain integrity threatened:
**                a)POKE to file chain length
**                b)POKE to 1st 8 nibbles of IRAM
**          ELSE
**             A(A), R0 = block start
**                   R1 = block end
**            D(A),C(A) = block size
**             Carry clear
**             P=0
**
** Calls:    PROTCK, FRSTFI, CNFFND, C=RAME, PKFIX1
**
** Uses:     A-D, R0-R3, D0, D1
**
** Detail:   Can RENAME a SECURE file, but can't POKE to ANY
**           part of a SECURE file
**
**           Can ATTEMPT a POKE to any unprotected (not SECURE
**            or PRIVATE) file, regardless of memory type -
**            ROM, EEPROM, etc; It is known that a POKE to
**            EEPROM will behave in an undesirable way
**
**           ADRPRT exits w/o error if block start=block end
**
** Stack lvls: 2
**
** History:
**
**    Date     Programmer   Modifications
**  --------   ----------   -------------
**  07/04/82   S.W.         Improved documentation
**  10/19/82   S.W.         Use symbolic bROMTB before
**                           calling CNFFND.
```

```
                        ** 07/22/83   S.W.        Don't allow POKE to MAINST.
                        **                         Add call to PKFIX1.
                        **
                        ************************************************************
                        ************************************************************
                        *
                        *
1C69F C0     ADRPRT A=A+B  A
1C6A1 101              R1=A                       PTR JUST PAST BLOCK END
1C6A4 D9               C=B    A
1C6A6 108              R0=C                       BLOCK START
1C6A9 D7               D=C    A
                        * If Block Start = Block End, nothing to check
1C6AB 8A2              ?A=C   A
1C6AE B5               GOYES  ADRDN1
                        * First guard against POKE to MAINST
1C6B0 8E00             GOSUBL =PKFIX1             errors on POKE to MAINST
      00
1C6B6 146              C=DAT0 A
1C6B9 7F11             GOSUB  FRSTFI
1C6BD 412              GOC    ADRP20
                        *
                        * D=BLOCK START; C=FILE END; B=FILE START
                        * SEE IF BLK END <= MAINST
1C6C0 142              A=DAT0 A
1C6C3 129              CR1EX                      C=BLOCK END; R1=FILE END
1C6C6 8BE              ?C<=A  A                   BLK END<=MAINST?
1C6C9 20               GOYES  ADRP17              SET CARRY IF YES
1C6CB 109    ADRP17 R1=C                          RESTORE BLOCK END
1C6CE 4A3              GOC    ADRDN1
                        * ENSURE THAT BLOCK START >= FILE START
1C6D1 DF               CDEX   A                   D=BLOCK END;C=BLOCK START
                        * B(A) POINTS TO FIRST AFFECTED FILE
1C6D3 D9               C=B    A
1C6D5 7EA0             GOSUB  PROTCK              SCAN THRU FILE CHAIN
1C6D9 461              GOC    ADRP30              REACHED E.O.F. CHAIN?
1C6DC 5A2              GONC   ADRDON              (B.E.T.)
                        *
                        * REACHED END OF MF FILE CHAIN - DIDN'T FIND BLK START
1C6DF 119    ADRP20 C=R1                          BLOCK END
1C6E2 D7               D=C    A
                        *      D1=(5) =RAMEND
                        *      C=DAT1 A
1C6E4 8F00             GOSBVL =C=RAME             C=(RAMEND)
      000
1C6EB 8BF              ?D<=C  A                   BLK CONTAINED IN SYS. RAM?
1C6EE 91               GOYES  ADRDON
                        *
1C6F0 32EF   ADRP30 LC(3)  =bROMTB
      B
1C6F5 8F00             GOSBVL =CNFFND
      000
1C6FC 137              CD1EX
1C6FF CA               A=A+C  A
1C701 102              R2=A                       SAVE END OF CFG CHAIN
```

```
482 1C704 501          GONC   ADRP45          (B.E.T.)
483               *
484 1C707 DB    ADRDON C=D    A
485 1C709 110   ADRDN1 A=RO                   BLK START
486 1C70C E2           C=C-A  A               #NIBS IN BLOCK
487 1C70E D7           D=C    A
488 1C710 01           RTN
489               *
490 1C712 137   ADRP40 CD1EX
491               *
492 1C715 8BA   ADRP45 ?A<=C  A               AT END OF CFG BUF?
493 1C718 FE           GOYES  ADRDON
494 1C71A 135          D1=C
495 1C71D 173          D1=D1+ 4               PT TO PLUG-IN ADDR
496 1C720 1573         C=DAT1 X               READ IN HIGH 3 NIBS
497 1C724 F2           CSL    A
498 1C726 F2           CSL    A               SHIFT IN ZEROES
499 1C728 175          D1=D1+ 6               PT. TO NEXT TABLE ENTRY
500 1C72B 8BF          ?C>=D  A               NO NEED TO SEARCH THIS CHAIN?
501 1C72E 4E           GOYES  ADRP40
502 1C730 133          AD1EX
503 1C733 103          R3=A                   SAVE PTR TO NEXT ENTRY
504               * C  IS AT START OF PORT - NOT AT START OF FILE CHAIN
505 1C736 135          D1=C
506 1C739 177          D1=D1+ 8               POINT PAST PROTECTED FIELD
507 1C73C 137          CD1EX
508 1C73F 135          D1=C
509 1C742 868          ?ST=0  8               PEEK?
510 1C745 A0           GOYES  ADRP50
511               * POKING AT LEAST 1 NIBBLE
512 1C747 110          A=RO                   BLOCK START
513 1C74A 8B2          ?A<C   A
514 1C74D 92           GOYES  POKERR          POKING INTO PROTECTED FIELD?
515               *   D1 IS NOW AT START OF FILE CHAIN
516 1C74F 118   ADRP50 C=RO
517 1C752 D7           D=C    A               BLOCK START
518 1C754 137          CD1EX                  RESTORE C(A)
519 1C757 7180         GOSUB  FRSTFI
520 1C75B 119          C=R1
521 1C75E D7           D=C    A               BLOCK END
522 1C760 4B0          GOC    ADRP54          Reached chain end before block end?
523               * B(A) POINTS TO 1ST AFFECTED FILE
524 1C763 D9           C=B    A
525 1C765 7E10         GOSUB  PROTCK
526 1C769 5D9          GONC   ADRDON          FOUND END OF BLOCK?
527 1C76C 11B   ADRP54 C=R3
528 1C76F 112          A=R2
529 1C772 42A          GOC    ADRP45          (B.E.T.)
530               *
531 1C775              BSS    1
532               *
```

```
                    EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:    PROTCK  -  Protection Check
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Ensures not PEEKing into PRIVATE file, not POKEing into
                **      PRIVATE or SECURE file, and not POKEing into any file's
                **      chain length field
                **
                ** Entry:
                **      P      = 0
                **      C(A) points to file chain start of affected chain
                **      D(A) is block end
                **      S8=1 => POKE operation
                **      R0   is block start
                **
                ** Exit:
                **      P      = 0
                **      Carry set =>  block extends beyond file chain
                **           clr =>  block ends within file chain
                **      D, R0, S8 preserved
                **
                **           Error exits (no return) if proposed
                **             operation violates file's protection (ePROT)
                **             or file chain integrity (eFACCS)
                **
                ** Calls:      GETPRO, FILSKP
                **
                ** Uses.......
                **  Exclusive: A(A), B(A), C(A)
                **  Inclusive: A(A), B(A), C(A), D1
                **
                ** Stk lvls:   1
                **
                ** NOTE:
                **      previously guarded against POKEing across file chain
                **      boundaries - that check was eliminated
                **
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **   --------   ----------   ------------------------------------
                **  07/05/82   S.W.         Added documentation
                **
                ********************************************************************
                ********************************************************************
                *
1C776 3100 =POKERR LC(2)  =eFACCS        Illegal file access
1C77A 8C00  mferr  GOLONG =MFERRj
          00
```

```
587                *
588 1C780 03     DONE   RTNCC
589                *
590 1C782 8BF    PROCK- ?D<=C  A              BLK END IS IN FILE JUST CHECKED?
591 1C785 BF            GOYES  DONE
592                *
593 1C787 135    PROTCK D1=C
594 1C78A 14B           A=DAT1 B
595 1C78D 968           ?A=0   B
596 1C790 00            RTNYES
597 1C792 D5            B=C    A
598 1C794 8F00          GOSBVL =GETPR1
          000
599 1C79B 4ED           GOC    mferr         PRIVATE?
600 1C79E 8F00          GOSBVL =FLSKPB       GET END OF FILE IN C
          000
601 1C7A5 868           ?ST=0  8             INTEND TO PEEK?
602 1C7A8 AD            GOYES  PROCK-        YES=>OK IF SECURE
603 1C7AA 832           ?SB=0                UNSECURE?
604 1C7AD A0            GOYES  PROTC3
605 1C7AF 3100 =RFPROT  LC(2)  =eFPROT
606 1C7B3 66CF  PROTC2  GOTO   mferr
607                * FILE START IN B; FILE END IN C; BLK START IN R0; BLK END IN R1
608                * IF DOING A POKE, ENSURES not POKEing into file length field
609 1C7B7 DA    PROTC3 A=C    A              SAVE PTR TO END OF FILE
610 1C7B9 D2           C=0    A
611 1C7BB 3102         LC(2)  =oFLENh
612 1C7BF C9           C=C+B  A              ADDR OF FILE CHAIN OFFSET
613 1C7C1 8BF          ?D<=C  A              BLK END<=FILE CHAIN OFFSET ADDR?
614 1C7C4 CB           GOYES  DONE
615 1C7C6 D5           B=C    A
616 1C7C8 D2           C=0    A
617 1C7CA 305          LC(1)  5
618 1C7CD C1           B=B+C  A              PT TO BGN OF NEXT FIELD IN HEADER
619 1C7CF 118          C=R0                 BLK START
620 1C7D2 8B1          ?C<B   A
621 1C7D5 1A           GOYES  POKERR
622 1C7D7 D6           C=A    A
623 1C7D9 58A          GONC   PROCK-        (B.E.T.)
624                *
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:    FRSTFI  -  First File
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Finds 1st file in chain (if any) affected by PEEK/POKE
                **
                ** Entry:
                **      P      =  0
                **      C(A) points to chain start
                **      D(A) is block start
                **
                ** Exit:
                **      P      =  0
                **      D(A) is preserved
                **      Carry set => No file in chain affected
                **                   D1 points to 00 byte at chain end
                **            clr => B(A) points to 1st affected file
                **                   C(A) points to end of 1st affected file
                **
                ** Calls:      FILSKP
                **
                ** Uses.......
                **  Exclusive: D1, A(B), B(A)
                **  Inclusive: A(A), B(A), C(A), D1
                **
                ** Stk lvls:   1
                **
                ** History:
                **
                **    Date     Programmer              Modification
                **  --------   ----------   ---------------------------------
                **  07/05/82   S.W.         Added documentation
                **
                ********************************************************************
                ********************************************************************
                *
1C7DC 135    FRSTFI D1=C
1C7DF 14B           A=DAT1 B
1C7E2 968           ?A=0   B
1C7E5 00            RTNYES
1C7E7 D5            B=C    A              Save pointer to file start
1C7E9 8F00          GOSBVL =FILSKP
      000
1C7F0 8BB           ?D>=C  A              BLK ST>=ST OF NXT FILE?
1C7F3 9E            GOYES  FRSTFI
1C7F5 01            RTN
                *
```

```
                        STITLE ADDR$
                ********************************************************************
                ********************************************************************
                **
                ** Name:     ADDR$   -  ADDR$ Function Execution
                **
                ** Category:   FNEXEC
                **
                ** Purpose:  ADDR$ function returns the address of the
                **           specified file.
                **
                ** Entry:    String on stack via EXPEXC
                **
                ** Exit:     Ascii string containing HEX address pushed on stk
                **           Exit via PCEXPR
                **             ERROR exits if:
                **              Illegal File spec - eFSPEC
                **              File not found    - eFnFND
                **
                ** Calls:    FILXQ, PREFLF, FINDF, STRHDR, REV$, HEXASC
                **           SAVD1, RSTD1, FTYPFD
                **
                ** Uses:     F-R0-0, A, B, C, D, D1, D0, S-R1-2, R0, R1, S6, S8
                **           F-R0-1
                **
                ** Detail:   Works for mainframe files (incl. plug-ins) only
                **           ADDR$(<string expr>)
                **
                ** Stack lvls: 4 (FILXQ$ can use 3)
                **
                ** History:
                **
                **    Date     Programmer   Modifications
                **  --------   ----------   -------------
                **  07/05/82   S.W.         Added documentation
                **  12/01/82   S.W.         ADDR$ returns pointer to file
                **                           header, not offset to data.
                **
                ********************************************************************
                ********************************************************************
                *
                *
1C7F7 8D00  AD$E30 GOVLNG =INVFSP        Invalid FILE SPEC
      000
                *
1C7FE 411          NIBHEX 411            STRING PARM
1C801 728D =ADDR$  GOSUB  SAVD0
1C805 8F00         GOSBVL =FILXQ$        RTNS FILE NAME IN A
      000
1C80C 5AE          GONC   AD$E30         ILLEGAL FILE SPECIFIER?
                *
1C80F 756D         GOSUB  SAVD1          SAVE D1 IN F-R0-1
1C813 8F00         GOSBVL =FINDF+
      000
1C81A 489          GOC    PROTC2         NOT FOUND?
```

```
728             * D1 CONTAINS ADDRESS OF FILE -
729 1C81D 133           AD1EX
730 1C820 727D          GOSUB  RSTD1          RESTORE D1
731 1C824 AF2   ADR$10  C=0    W              WANT C(S)=0;C(A)=0
732 1C827 24            P=     4
733 1C829 80FF          CPEX   15             C(S)=4;P=0
734 1C82D AF7           D=C    W              D(S)=C(S)=#NIBS-1 TO CONVRT;D(A)=0
735 1C830 8E00          GOSUBL =HEXASC        CONVERT TO ASCII
          00
736 1C836 B47           D=D+1  S              #NIBS THAT WERE CONVERTED
737 1C839 A47           D=D+D  S              #OF NIBS TO PUSH ON STACK
738 1C83C 813           DSLC
739 1C83F DB            C=D    A
740 1C841 8F00          GOSBVL =STRHDR        PUT HEADER ON STACK
          000
741 1C848 80D0          P=C    0
742 1C84C 0D            P=P-1
743 1C84E AF4           A=B    W              ONLY NEED WP, BUT LESS CODE
744 1C851 1511          DAT1=A WP
745 1C855 1CF           D1=D1- 16
746 1C858 601D          GOTO   PK$70          RESTORE PC, ETC.
747             *
748             *
```

```
749                     STITLE DTH$ Function Execution
750                *****************************************************************
751                *****************************************************************
752                **
753                ** Name:     HEX$     -  DTH$ Function Execution
754                **
755                ** Category:   FNEXEC
756                **
757                ** Purpose:    DTH$ function that converts a decimal number
758                **             to an ascii hexadecimal address
759                **
760                ** Entry:      Numeric expression on stack - pointed to by D1
761                **
762                ** Exit:       String pushed on stack - exit via EXPR
763                **               ERROR exits if:
764                **                 Numeric expr is complex - eDATTY
765                **                 Numeric expr negative or too large - eARGOR
766                **
767                ** Calls:    POPNUM, HEXASC, STRHDR, REV$
768                **
769                ** Stack lvls: 3
770                **
771                ** Uses:     A-D, R1, SB, D1
772                **
773                ** Detail:   HEX$ <numeric expression>
774                **
775                ** History:
776                **
777                **    Date     Programmer   Modifications
778                **  --------   ----------   -------------
779                **  07/05/82   S.W.         Added documentation
780                **
781                *****************************************************************
782                *****************************************************************
783                *
784 1C85C 811          NIBHEX 811
785 1C85F 7B8C =HEX$   GOSUB  POPNUM
786 1C863 40C          GOC    ADR$10         (B.E.T.)
787                *
```

```
788                     STITLE HTD Function Execution
789                *****************************************************************
790                *****************************************************************
791                **
792                ** Name:    HXDEC   -  HTD Function Execution
793                **
794                ** Category:   FNEXEC
795                **
796                ** Purpose:    HTD function which converts an ascii repre-
797                **             sentation of a hex address to a decimal number
798                **
799                ** Entry:      String on stack & pointed to by D1
800                **
801                ** Exit:       Decimal number pushed on stack
802                **
803                ** Calls:      ADDRCK, HDFLT
804                **
805                ** Stack lvls: 4
806                **
807                ** Uses:     A,B,C(A),D(A), D1,D0
808                **
809                ** Detail:   This function and DTH$ are useful with ADDR$.
810                **           HTD <string expression>
811                **
812                ** History:
813                **
814                **    Date     Programmer   Modifications
815                **  --------   ----------   ------------
816                **  07/05/82   S.W.         Improved documentation.
817                **                           Modified code to use AVMEME to
818                **                           push on stack (to not crash into
819                **                           SAVE area - was using TFORN)
820                **
821                *****************************************************************
822                *****************************************************************
823                *
824                *
825 1C866 411            NIBHEX 411
826 1C869 783D =HXDEC   GOSUB  ADDRCK
827 1C86D D4            A=B    A
828 1C86F 1CF           D1=D1- 16
829 1C872 8D00          GOVLNG =LEN20      .     Calls HDFLT, pushes result
          000
830                *
831                *
832 1C879               END
```

```
 AD$E30   Abs  116727 #1C7F7 -   718   723
=ADDR$    Abs  116737 #1C801 -   721
 ADDRCK   Abs  116133 #1C5A5 -   213   117   336   826
 ADR$10   Abs  116772 #1C824 -   731   786
 ADRCK2   Abs  116183 #1C5D7 -   231   234
 ADRDN1   Abs  116489 #1C709 -   485   445   459
 ADRDON   Abs  116487 #1C707 -   484   466   475   493   526
 ADRP17   Abs  116427 #1C6CB -   458   457
 ADRP20   Abs  116447 #1C6DF -   469   450
 ADRP30   Abs  116464 #1C6F0 -   477   465
 ADRP40   Abs  116498 #1C712 -   490   501
 ADRP45   Abs  116501 #1C715 -   492   482   529
 ADRP50   Abs  116559 #1C74F -   516   510
 ADRP54   Abs  116588 #1C76C -   527   522
 ADRPRT   Abs  116383 #1C69F -   438   122   354
 ASCHEX   Abs  116195 #1C5E3 -   272   231   368
 ASCHX2   Abs  116204 #1C5EC -   275   287
 ASCHX5   Abs  116218 #1C5FA -   281   273
 C=RAME   Ext                -   473
 CNFFND   Ext                -   478
 CONVUC   Ext                -   281
 DONE     Abs  116608 #1C780 -   588   591   614
 DRANGE   Ext                -   272
 EXPEX-   Ext                -   342
 ExpExc   Ext                -   335
 F-RO-0   Abs  194715 #2F89B -    11   159
 F-RO-1   Abs  194720 #2F8A0 -    11   154   163
 FILSKP   Ext                -   671
 FILXQ$   Ext                -   722
 FINDF+   Ext                -   726
 FIXDC    Ext                -   333
 FLSKPB   Ext                -   600
 FLTDH    Ext                -    56
 FRSTFI   Abs  116700 #1C7DC -   666   449   519   673
 GETPR1   Ext                -   598
=HEX$     Abs  116831 #1C85F -   785
 HEXASC   Ext                -   140   735
=HXDEC    Abs  116841 #1C869 -   826
 INVARG   Abs  116249 #1C619 -   289   109   217   221   284
 INVFSP   Ext                -   718
 InvArg   Ext                -   289
 LEN20    Ext                -   829
 MFERRJ   Ext                -   586
 PCEXPR   Ext                -   151
=PEEK$    Abs  115981 #1C50D -   112
 PK$20    Abs  116036 #1C544 -   136   144
 PK$60    Abs  116067 #1C563 -   147   137
 PK$70    Abs  116073 #1C569 -   149   746
 PK$E24   Abs  115973 #1C505 -   109   115
 PKFIX1   Ext                -   447
=POKE     Abs  116265 #1C629 -   335
 POKE10   Abs  116347 #1C67B -   363   372
=POKE25   Abs  116377 #1C699 -   374   364
 POKEP    Ext                -   334
=POKERR   Abs  116598 #1C776 -   585   514   621
```

POP1R	Ext		-	53		
POP1S	Ext		-	345		
POPNUM	Abs	115950	#1C4EE	-	53	112	785
PROCK-	Abs	116610	#1C782	-	590	602	623
PROTC2	Abs	116659	#1C7B3	-	606	727	
PROTC3	Abs	116663	#1C7B7	-	609	604	
PROTCK	Abs	116615	#1C787	-	593	464	525
RANGE	Ext			-	283		
REN180	Ext			-	374		
REV$	Ext			-	150		
REVPOP	Ext			-	213		
=RFPROT	Abs	116655	#1C7AF	-	605		
=RSTD1	Abs	116118	#1C596	-	163	130	730
S-RO-0	Abs	194673	#2F871	-	11	338	350
=SAVD0	Abs	116103	#1C587	-	158	55	721
=SAVD1	Abs	116088	#1C578	-	153	118	725
STRHDR	Ext			-	132	740	
TFORN	Abs	193950	#2F59E	-	11	360	
bROMTB	Abs	3070	#00BFE	-	12	477	
eFACCS	Ext			-	585		
eFPROT	Ext			-	605		
mferr	Abs	116602	#1C77A	-	586	599	606
oFLENh	Abs	32	#00020	-	12	611	

Input Parameters

  Source file name is SG&POK::MS

  Listing file name is SG/POK:TI:ML::-1

  Object file name is SG%POK:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    M   M  N   N   &      CCC   DDDD
 2            *    MM MM  N   N  & &    C   C  D  D
 3            *    M M M  NN  N  & &    C      D  D
 4            *    M M M  N N N   &     C      D  D
 5            *    M   M  N  NN  & & &  C      D  D
 6            *    M   M  N   N  &  &   C   C  D  D
 7            *    M   M  N   N   && &   CCC   DDDD
 8            *
 9                    TITLE  Card Reader Module <831212.1617>
10 1C879             ABS    #1C879
11             TEST   EQU    1              Test version for first chip
12            *
13                    RDSYMB SB%RAM::MS
14                    RDSYMB SB%KCM::MS
15                    RDSYMB TI%EQU::MS
16            *
17             fTMOUT EQU    3              TIMEOUT TIMER FLAG
18             LIF1R  EQU    #498C
19             LIF1RT EQU    #49
20             HDRSIZ EQU    37
21            *!!
22            *!! Changes needed to implement enhancement request #993-6
23            *!! are indicated with *!! comment lines
24            *!!
```

```
                STITLE Copy to card
*****************************************************************
*****************************************************************
**
** Name:(S) FILCRD  -  Copy File To Card
**
** Category:   FILUTL
**
** Purpose:
**      Copy file from memory to card.
**
** Entry:
**      C[A] points to start of file header.
**      R1 contains name to be used on card.  Zeroes if no
**        name specified (use name of file).
**      S8=1 if private card requested.
**
** Exit:
**      Returns if write completed.
**      NXTSTM if write aborted.
**      Error exits:
**
** Calls:       ALIGN, BLANKC, CHKSUM, CMPTIM, CR??, CRDOFF,
**              CSLW5, D1+13B, D1+21B, D1+29B, DAYYMD, FNDPRT,
**              FROMDT, FTYPF#, IMPFLD, IOAL36, LCTRKS, MAXTRK,
**              POLL, PREPDT, PREPHD, R1TODO, RCO1, RD8SV,
**              RDSOC, RDYTRK, READCS, READFL, RTODP, STO1,
**              STDRG?, TOCARD, TODT, TRKDON, VFYCRD, WAITM+,
**              WRIT8S, WRITE, WRITFL, WRMSG, WRT2-0, YMDDAY,
**              aslw5, asrw5, crlfnd, csrw5, fpoll, idiva,
**              noscrl.
**
** Uses.......
**              A,B,C,D,P,D0,D1,ST,R0-R4,
**              SNAPBF, SAVSTK, SCRTCH, 8 levels in RSTKBF.
**
** Stk lvls:   4
**
** Detail:
**      Card format chosen for compatibility with HP-75.
**      Card is divided into four fields, each preceded by a
**      hardware recognized flag and followed by a zero byte.
**      Fields are separated by a 66 ttfc (timetrack flux
**      change) gap.  fields are as follows:
**
** ====   Start-of-Card: recorded at factory when timetrack
**                       is recorded.
**          SOC marker: "HP"  (2)
**          format: "CV"  (2)
**          size: # bytes available after write-protect field
**                (specific to Corvallis format)  (2)
**                for 10" cards: 2BC (=700 base 10)
**          (reserved): 0000 (2)
**
** ====   Write-protect: 4-byte field:
```

```
80          **          0000 for write-enabled cards.   \ (2)
81          **          FFFF for write-protected cards. /
82          **          (reserved): 0000 (2)
83          **          padding added by HP-75  (1)
84          **
85          ** ====   Data Header: identifies file, contains security
86          **                     information.
87          **
88          **       % identifies fields which differ between HP-71 and
89          **         HP-75 format.  HP-75 format is only used
90          **         for LIF1 (text) files.
91          **
92          **     % 0:  sub-format (1): 00 for LIF1 file (HP-75 subformat)
93          **                           01 for HP-71 files (HP-71 subfmt)
94          **         1:  track# (1)
95          **         2:  # of tracks in set (1)
96          **         3:  # bytes in this track (2)
97          **         5:  # bytes in file (2)
98          **     % 7:  file type (2): HP-75 filetype (HP-75 subformat)
99          **                          LIF filetype (HP-71 subformat)
100         **         9:  creation date (4): hex seconds since start of
101         **                                century.
102         **         13: file name (8)
103         **     % 21: password (4): blanks for LIF1 filetype (HP-75)
104         **           implementation (4): (HP-71 subformat)
105         **         25: marker (2): checksum of entire file, including
106         **                         file header.
107         **         27: partial statement status (1)
108         **         28: s1 (2)
109         **         30: s2 (2)
110         **         32: data checksum (2): 2-byte checksum of data field.
111         **         34: header checksum (1): 2-byte sum of header field,
112         **                                  folded to one byte without
113         **                                  wraparound carry.
114         **         35: (reserved) (1)
115         **             padding added by HP-75 (2)
116         **
117         **       File headers for the two subformats differ only in
118         **       bytes 0, 7-8 and 21-24.
119         **
120         ** ====   Data: 650 bytes
121         **
122         **               padding added by HP-75 (3)
123         **
124         **
125         **
126         **       All files except LIF1 will use LIF filetype in the
127         **       filetype.  For the curious among you, HP-75
128         **       filetypes consist of two bytes:
129         **
130         **         high order byte: 00=HP-75 system file
131         **                          ??=HP-75 text file
132         **                          ??=HP-75 basic file
133         **                          ??=HP-75 appointment file
134         **                          ??=HP-75 lex file
```

```
135           **                        ??=HP-75 keds file
136           **                        "I"=LIF1
137           **
138           **       low order byte: HP-75 attribute byte.  Identifies
139           **               file capabilities.  bit masks as defined by
140           **               HP-75 are:
141           **                 80=in rom
142           **                 40=file runnable
143           **                 20=file editable
144           **                 10=file listable
145           **                 08=file purgable
146           **                 04=file copyable
147           **                 02=standard lined file
148           **                 01=token file
149           **         two important bit masks are:
150           **                 34=private file
151           **                 7E=data file for print#/read#
152           **
153           **         HP-75 documentation identifies some basic file
154           **         types:
155           **               0062=calculator file
156           **               0000=system file
157           **               013E=text file
158           **               027E=basic file
159           **               030C=appointment file
160           **               050C=alpd fild
161           **               0008=diagnostic file
162           **
163           **
164           ** ----------
165           **
166           ** Function of scratch registers in card write:
167           **   R0=scratch
168           **   R1[A]=file pointer
169           **   R1[9-5]=amount of zero-padding at end of file in bytes
170           **            (used to bring LIF1 to sector boundary).
171           **   R2=pointer to I/Obuffer containing header
172           **
173           ** Function of status bits:
174           **   S1: Used by Verify to suppress DATA ERROR in read from
175           **       FIFO.
176           **   S2: Indicate we are on last track of card set.
177           **
178           ** Algorithm:
179           **      Check for presence of card reader; eDVCNF if absent.
180           **      Allocate I/O buffer for building header; eMEM if
181           **        no room for buffer.
182           **      Fetch filetype from file.  If not copying to PCRD then
183           **        goto 2.
184           **      Search for filetype in filetype table.  If found then
185           **        goto 1.
186           **      If filetype not in standard range then goto 2.
187           **      Set privacy bit in filetype.  Goto 2.
188           **    1: If there are < 3 entries in filetype table for this
189           **        filetype then goto 2.
```

```
190             **      Read third entry (private) from filetype table.
191             **  2:  Store filetype in card header I/Obuffer.
192             **      Store passed destfile name in header I/Obuffer.
193             **      Compute time (seconds since start of century) and store
194             **        in header I/Obuffer.
195             **      If we are writing out LIF1 filetype then write HP-75
196             **        LIF1 filetype to filetype field and 00 to subformat
197             **        field, else write 01 to subformat field.
198             **      Write 01 to track# field in header I/Obuffer.
199             **      If copycode=8 then poll for somebody to copy card;
200             **        eFYPE if not handled.
201             **      Compute file length in bytes: (chain length-5)/2 if
202             **        copycode#1, (chain length-13)/2 if copycode = 1;
203             **        rounded up to byte.  If > FFFF bytes then eF2BIG.
204             **      If filetype = LIF1 then pad file length up to sector
205             **        boundary (256 bytes).
206             **      Write file length to header I/Obuffer.
207             **      Compute implementation field, write to header
208             **        I/Obuffer.
209             **      Perform checksum of entire file for "marker" byte--byte
210             **        which uniquely identifies card set.  Write to
211             **        header I/Obuffer.
212             **      Compute # tracks in card set.  Write to header
213             **        I/Obuffer.
214             **  3:  Read track# and maxtrack# from header I/Obuffer.
215             **        Deallocate buffer and return if track# > maxtrack#.
216             **      Compute trksize.  Write to header I/Obuffer.
217             **      Compute checksum of this track.  Write to header
218             **        I/Obuffer.
219             **      Write 0's to partial card recovery fields (since
220             **        recovery is not implemented).
221             **      Compute header checksum.  Write to header I/Obuffer.
222             **      Perform WCRD poll.
223             **  4:  Prompt "Wrt: Align then ENDLN" and wait for ENDLINE or
224             **        ATTN or f-ATTN or timeout.
225             **      If ATTN or f-ATTN or timeout then abort.
226             **      Prompt "Pull xxx of xxx".
227             **      {card now starts moving.}
228             **      Verify start-of-card (SOC) field.  If wrong then
229             **        eUNKCD and goto 4.
230             **      Read write-protect field.  If not 0's then ePROTD and
231             **        goto 4.
232             **      Switch to write mode.  Write 16 nibbles of 0's.
233             **      Write BREAK to card.
234             **      Write header I/Obuffer to card.
235             **      Write 16 nibbles of 0's to card.
236             **      Write BREAK to card.
237             **      Write data field to card, padding with 0's as necessary
238             **        for text files.
239             **      Turn off card reader.
240             **  5:  Prompt "Vfy: Align then ENDLN" and wait for ENDLN.
241             **      Prompt "Pull xxx of xxx".
242             **      Verify SOC field.  If fail, eUNKCD and goto 5.
243             **      Skip write-protect field.
244             **      Verify header field.  If error, eVFYER and goto 4.
```

```
245                **      Verify data field.  If error, eVFYER and goto 4.
246                **      Turn off card reader.
247                **      Update file pointer and increment track#.  Goto 3.
248                **
249                ** History:
250                **
251                **     Date     Programmer            Modification
252                **   --------   ----------   ---------------------------------
253                **   07/12/82   NM           Added documentation
254                **   02/25/83   NM           Updated "CALLS" section
255                **
256                ********************************************************************
257                ********************************************************************
258                 TRKSIZ EQU    650           #bytes in data field
259 1C879 129  =FILCRD CR1EX                    Save file pointer
260 1C87C 10B         R3=C                     Save filename
261 1C87F 7116        GOSUB  CR??              Check for card reader present
262                * Build header in i/obuffer
263 1C883 7666        GOSUB  IOAL36            Allocate buffer for header
264 1C887 460         GOC    FILC05            Go if successful
265 1C88A 6CC6        GOTO   NROOM             Insufficient memory
266 1C88E 137   FILC05 CD1EX
267 1C891 10A         R2=C                     Save buffer pointer
268 1C894 7025        GOSUB  RTODP
269                * Filetype: bytes 7-8
270 1C898 16F         D0=D0+ 16                Point at filetype
271 1C89B D0          A=0    A
272 1C89D 15A3        A=DATO 4                 Read filetype from file
273 1C8A1 17D         D1=D1+ 2*7               Point at filetype field in hdr
274 1C8A4 868         ?ST=0  8                 Privacy requested?
275 1C8A7 73          GOYES  FTYP40            No.  use virgin filetype
276 1C8A9 8F00        GOSBVL =FTYPF#           Do we recognize filetype?
            000
277 1C8B0 431         GOC    FTYP30            Yes
278 1C8B3 72A5        GOSUB  STDRG?            No.  filetype in std range?
279 1C8B7 462         GOC    FTYP40            No.  No private.  Use orig type.
280 1C8BA 302         LCHEX  2                 Std range.  set bit#1
281 1C8BD 0E0E        A=A!C  P                 Makes private
282 1C8C1 5C1         GONC   FTYP40            B.E.T.
283 1C8C4 134   FTYP30 D0=C                    Point at table entry
284 1C8C7 16E         D0=D0+ 15
285 1C8CA 1524        A=DATO S                 #entries in table
286 1C8CE 2F          P=     15
287 1C8D0 303         LCHEX  3
288 1C8D3 982         ?A<C   P                 Are there three entries?
289 1C8D6 80          GOYES  FTYP40            No.  less.  cannot make private.
290 1C8D8 168         D0=D0+ 9                 Yes.  point at third entry
291 1C8DB 142         A=DATO A                 Read private filetype
292 1C8DE 1593  FTYP40 DAT1=A 4                Write out filetype
293 1C8E2 72D4  FTYP50 GOSUB  RTODP            Restore pointers
294                * File name: bytes 13-20
295 1C8E6 7BF5        GOSUB  D1+13B            Point at name in header
296 1C8EA 113         A=R3                     Retrieve name passed to me
297 1C8ED 1517        DAT1=A W
298                * Creation date: bytes 9-12
```

```
299                * Truncating year to lower 2 digits will put us in
300                *   leap-century.
301 1C8F1 8F00         GOSBVL =STO1          Stash R0 & R1
          000
302 1C8F8 8F00         GOSBVL =CMPTIM        Compute time
          000
303 1C8FF 8F00         GOSBVL =TODT          To date and time-of-day
          000
304 1C906 AF9          C=B    W
305 1C909 108          R0=C                  Stash time-of-day
306 1C90C AF6          C=A    W              Day#
307 1C90F 8F00         GOSBVL =DAYYMD        To y,m,d
          000
308 1C916 AD0          A=0    M
309 1C919 AA0          A=0    XS             Truncate year
310 1C91C 8F00         GOSBVL =YMDDAY        To day#
          000
311 1C923 AFA          A=C    W
312 1C926 118          C=R0
313 1C929 8F00         GOSBVL =FROMDT        To time in secs
          000
314 1C930 8F00         GOSBVL =RCO1          Restore R0 & R1
          000
315 1C937 7D74         GOSUB  RTODP
316 1C93B 178          D1=D1+ 9
317 1C93E 178          D1=D1+ 9              Point at date
318 1C941 15D7         DAT1=C 8              Write it out
319                * Subformat and track#: bytes 0-1
320 1C945 1C3          D1=D1- 2*2            Point at filetype field
321 1C948 143          A=DAT1 A
322 1C94B D2           C=0    A
323 1C94D E6           C=C+1  A              Lif1 filetype
324 1C94F 23           P=     3
325 1C951 916          ?A#C   WP             Is this lif1 file?
326 1C954 01           GOYES  FILC07         No.  subformat=01
327 1C956 20           P=     0
328 1C958 33C8         LC(4)  LIF1R          HP-75 lif1 filetype
          94
329 1C95E 15D3         DAT1=C 4              Write to header
330 1C962 D2           C=0    A              Subformat=00
331 1C964 B26   FILC07 C=C+1  XS             Track#=01
332 1C967 1CD          D1=D1- 2*7            Point at subformat
333 1C96A 15D3         DAT1=C 4              Write subformat & track#
334                * File length: bytes 5-6
335 1C96E 7C44         GOSUB  R1TODO
336 1C972 16F          D0=D0+ 16
337 1C975 164          D0=D0+ 5              Point at flags hinib
338 1C978 1524         A=DAT0 S              Read copy code
339 1C97C 16A          D0=D0+ 11
340 1C97F 142          A=DAT0 A              Read file chain length
341 1C982 D2           C=0    A
342 1C984 2F           P=     15
343 1C986 3115         LCHEX  51             A[A]=5 (Chlen fld size),A[S]=1
344 1C98A 946          ?A#C   S              Copy code=1?
345 1C98D 60           GOYES  FILC10         No
```

```
346 1C98F 310D          LCHEX  D0            A[A]=13 (Chlen + impfld size)
347 1C993 A44   FILC10  A=A+A  S             Copy code=8?
348 1C996 5F1          GONC   FILC12         No
349 1C999 18F          D0=D0- 16             Yes.  point at filetype
350 1C99C 15A3         A=DAT0 4
351 1C9A0 8F00         GOSBVL =POLL          Poll for card write routine
          000
352 1C9A7 42           CON(2) =pWCRD8
353 1C9A9 831          ?XM=0                 Was it handled?
354 1C9AC 60           GOYES  FILCAB         Yes.  finish up.
355 1C9AE 60B5         GOTO   FTYPER         Error out
356 1C9B2 6055  FILCAB GOTO   BUFDAL
357 1C9B6 D8    FILC12 B=A    A              Hold chain len
358 1C9B8 AF0          A=0    W
359 1C9BB 132          ADOEX                 Fetch pointer to chain len
360 1C9BE 130          D0=A
361 1C9C1 CA           A=A+C  A              Point at data field
362 1C9C3 101          R1=A                  Store new file pointer
363 1C9C6 D4           A=B    A              Get chain length
364 1C9C8 EA           A=A-C  A              Size of data field
365 1C9CA E4           A=A+1  A              For rounding
366 1C9CC 81C          ASRB                  Size of data field in bytes
367 1C9CF 24           P=     4
368 1C9D1 908          ?A=0   P              Size <= FFFF bytes?
369 1C9D4 A0           GOYES  FILC14         Yes
370 1C9D6 1D00         D1=(2) =eF2BIG
371 1C9DA 6E56         GOTO   errout         Error out
372 1C9DE 14F   FILC14 C=DAT1 B              Read subformat
373 1C9E1 A6E          C=C-1  B              Set carry if subformat=0
374 1C9E4 D6           C=A    A              Copy filesize in bytes.
375 1C9E6 129          CR1EX
376 1C9E9 8EC1         GOSUBL aslw5          Doesn't affect carry.
          80
377 1C9EF DA           A=C    A
378 1C9F1 121          AR1EX                 Filesize to R1[9-5] and A[A].
379 1C9F4 5C0          GONC   FILC15         Go if not LIF1 file.
380 1C9F7 CC           A=A-1  A              Decrement A[2-4] if A[B]=00.
381 1C9F9 AE0          A=0    B              Must pad to sector boundary.
382 1C9FC A6C          A=A-1  B
383 1C9FF E4           A=A+1  A
384 1CA01 179   FILC15 D1=D1+ 2*5            Point at file length field
385 1CA04 1593         DAT1=A 4              Write out
386 1CA08 20           P=     0
387 1CA0A D2           C=0    A
388 1CA0C 3102         LC(2)  32
389 1CA10 C9           C=B+C  A              #nibs to sum for marker
390 1CA12 18F          D0=D0- 16
391 1CA15 18F          D0=D0- 16             Point at start of file header
392 1CA18 8F00         GOSBVL =CSLW5
          000
393 1CA1F 136          CDOEX
394 1CA22 108          R0=C                  Stash info for marker computation
395             * Implementation field: bytes 21-24
396 1CA25 134          D0=C                  Restore file hdr pointer
397 1CA28 1C9          D1=D1- 2*5
```

```
 398 1CA2B 14B          A=DAT1 B            Read subformat
 399 1CA2E 70B4         GOSUB  D1+21B        Point at imp field in card hdr
 400 1CA32 96C          ?A#0   B            Subformat=0?
 401 1CA35 E0           GOYES  FILC16       No
 402 1CA37 20           P=     0            Yes.  use password=blanks
 403 1CA39 8F00         GOSBVL =BLANKC
           000
 404 1CA40 560          GONC   FILC18       B.E.T.
 405 1CA43 7313  FILC16 GOSUB  IMPFLD       Compute implementation field
 406 1CA47 15D7  FILC18 DAT1=C 8            Write out imp field
 407                 * Marker: bytes 25-26
 408 1CA4B 118          C=R0
 409 1CA4E 134          D0=C                Point at start of file hdr
 410 1CA51 7FA7         GOSUB  csrw5        #nibs to checksum
 411 1CA55 7593         GOSUB  CHKSUM       Compute marker
 412 1CA59 177          D1=D1+ 2*4          Point at marker field
 413 1CA5C 1593         DAT1=A 4
 414                 * #tracks in set: byte 2
 415 1CA60 7502         GOSUB  MAXTRK
 416 1CA64 1C5          D1=D1- 2*3          Point at #tracks field
 417 1CA67 149          DAT1=A B
 418 1CA6A 842          ST=0   2            Indicate haven't started padding.
 419                 * Next: start card write loop
 420 1CA6D 7743  FILC40 GOSUB  RTODP
 421 1CA71 171          D1=D1+ 2            Point at trk#
 422 1CA74 D0           A=0    A
 423 1CA76 147          C=DAT1 A
 424 1CA79 AEA          A=C    B            Current track#
 425 1CA7C F6           CSR    A
 426 1CA7E F6           CSR    A
 427 1CA80 9EA          ?C>=A  B            Max < current track#?
 428 1CA83 E0           GOYES  FILC45       No
 429 1CA85 7544  FINCRD GOSUB  crlfnd       Clear display
 430 1CA89 7844         GOSUB  noscrl
 431 1CA8D 6574         GOTO   BUFDAL       Yes... done.  deallocate & rtn.
 432 1CA91 B6A   FILC45 A=A-C  B            Current trk#-last trk#
 433 1CA94 7513         GOSUB  LCTRKS       Size of full data field
 434 1CA98 177          D1=D1+ 2*4          Point at filesize
 435 1CA9B 96C          ?A#0   B            Is this last track?
 436 1CA9E 31           GOYES  FILC50       No
 437 1CAA0 15B3         A=DAT1 4            Yes, read filesize
 438 1CAA4 7AA3         GOSUB  idiva        Filesize mod trksize to b
 439 1CAA8 8AE          ?C#0   A            Mod=0?
 440 1CAAB 60           GOYES  FILC50       No, this is tracksize
 441 1CAAD 7CF2         GOSUB  LCTRKS       Yes, final card is full
 442 1CAB1 1C3   FILC50 D1=D1- 2*2          Point at trksize
 443 1CAB4 15D3         DAT1=C 4            Write #bytes in this track
 444 1CAB8 111          A=R1
 445 1CABB D8           B=A    A            Hold data pointer
 446 1CABD 7157         GOSUB  asrw5        #bytes left to write
 447 1CAC1 862          ?ST=0  2            Have we started padding?
 448 1CAC4 40           GOYES  FILC60       No.
 449 1CAC6 D0           A=0    A            Yes.  This whole track is pad.
 450 1CAC8 EA    FILC60 A=A-C  A            Valid data left to write.
 451 1CACA 570          GONC   FILC70       Go if entire track data.
```

```
452 1CACD 852          ST=1   2            We will pad.
453 1CAD0 F8           A=-A   A            This much padding.
454 1CAD2 7537  FILC70 GOSUB  aslw5
455 1CAD6 D4           A=B    A
456 1CAD8 101          R1=A                Restore.
457 1CADB 7BB1         GOSUB  PREPDT       Prepare for checksum calculations
458 1CADF 7B03         GOSUB  CHKSUM       Checksum of this track
459 1CAE3 75F3         GOSUB  D1+29B       (d1 still at trksize)
460 1CAE7 1593         DAT1=A 4            Write data checksum
461 1CAEB 7BA2         GOSUB  FNDPRT       Compute PSS information
462 1CAEF 112          A=R2
463 1CAF2 130          D0=A                Point to header
464 1CAF5 20           P=     0
465 1CAF7 D2           C=0    A
466 1CAF9 3144         LC(2)  2*34         Length header thru data cksum
467 1CAFD CA           A=A+C  A
468 1CAFF 131          D1=A                Point at header cksum field
469 1CB02 78E2         GOSUB  CHKSUM       Compute 2-byte checksum of hdr
470 1CB06 D6           C=A    A
471 1CB08 B94          ASR    WP
472 1CB0B BB4          ASR    X
473 1CB0E A6A          A=A+C  B            Compress to 1 byte
474 1CB11 1593         DAT1=A 4            Write hdr cksum & zeroes
475 1CB15 7214         GOSUB  fpoll        Card write poll
476 1CB19 53           CON(2) =pWCRD
477 1CB1B 1D00  FILC75 D1=(2) =eWALGN      "w: align card, hit key"
478 1CB1F 7D25         GOSUB  ALIGN
479 1CB23 560          GONC   FILC76       Go if no abort
480 1CB26 6293         GOTO   RTNABT
481 1CB2A 7934  FILC76 GOSUB  RDYTRK       "rdy (trk#) of (#trks)"
482 1CB2E 7E33         GOSUB  RDSOC        Get past soc stuff
483 1CB32 48E          GOC    FILC75       Error
484 1CB35 23    FILC90 P=     3
485 1CB37 918          ?A=0   WP           Write-protect field = 0?
486 1CB3A E0           GOYES  FIL120       Yes
487 1CB3C 1D00         D1=(2) =ePROTD
488 1CB40 7474         GOSUB  WRMSG        No.  "Protected"
489 1CB44 66DF  FIL100 GOTO   FILC75
490 1CB48 2F    FIL120 P=     15
491 1CB4A 301          LCHEX  1
492 1CB4D 75A5         GOSUB  READCS       Read fifth byte
493 1CB51 42F          GOC    FIL100
494 1CB54 7635         GOSUB  WRITE        Ok.  switch to write mode.
495 1CB58 7A96         GOSUB  WRT2-0       Write 16 zeroes
496 1CB5C 47E          GOC    FIL100
497 1CB5F 7E55         GOSUB  WRITFL       Set flgsrv
498 1CB63 40E          GOC    FIL100
499 1CB66 7021         GOSUB  PREPHD       Prepare to write header
500 1CB6A 7B41         GOSUB  TOCARD       Copy to card
501 1CB6E 45D          GOC    FIL100
502 1CB71 7186         GOSUB  WRT2-0       Field separator of zeroes
503 1CB75 4EC          GOC    FIL100
504 1CB78 7545         GOSUB  WRITFL       Set flgsrv
505 1CB7C 47C          GOC    FIL100
506 1CB7F 7711         GOSUB  PREPDT       Prepare to write data
```

```
507 1CB83 7231         GOSUB  TOCARD        Transfer to card
508 1CB87 4CB  FIL130  GOC    FIL100        Error
509 1CB8A 119          C=R1
510 1CB8D 7376         GOSUB  csrw5         Prepare for 0's at end.
511 1CB91 872          ?ST=1  2             Last track?
512 1CB94 40           GOYES  FIL140        Yes.  This is #bytes to pad.
513 1CB96 D2           C=0    A             No.  Pad zero bytes.
514 1CB98 81E  FIL140  CSRB
515 1CB9B 81E          CSRB                 Will write #mod4 + 4 bytes of 0.
516 1CB9E D7           D=C    A             Hold counter.
517 1CBA0 AF0          A=0    W             For write.
518 1CBA3 7245 FIL150  GOSUB  WRIT8S        Write
519 1CBA7 4FD          GOC    FIL130        Go if fail.
520 1CBAA CF           D=D-1  A             Decrement counter.
521 1CBAC 56F          GONC   FIL150        Back for more.
522 1CBAF 7585         GOSUB  WAITM+
523 1CBB3 7CF4         GOSUB  CRDOFF        Done writing
524 1CBB7 1D00 VFY000  D1=(2) =eVALGN       "v: align card, hit key"
525 1CBBB 7194         GOSUB  ALIGN
526 1CBBF 560          GONC   VFY010
527 1CBC2 66F2         GOTO   RTNABT
528 1CBC6 7D93 VFY010  GOSUB  RDYTRK
529 1CBCA 72A2         GOSUB  RDSOC         Get past soc stuff
530 1CBCE 48E          GOC    VFY000        Go if error
531 1CBD1 75E4         GOSUB  READFL        Set flgsrv for header field
532 1CBD5 71B0         GOSUB  PREPHD        Prepare to compare hdr field
533 1CBD9 7811         GOSUB  VFYCRD        Verify card
534 1CBDD 49D          GOC    VFY000
535 1CBE0 831          ?XM=0                Verify successfully?
536 1CBE3 E0           GOYES  VFY040        Yes
537 1CBE5 1D00 VFY035  D1=(2) =eVFYER
538 1CBE9 7BC3         GOSUB  WRMSG         "verify error"
539 1CBED 6D2F         GOTO   FILC75        Try again
540 1CBF1 75C4 VFY040  GOSUB  READFL        Set flgsrv for data field
541 1CBF5 71A0         GOSUB  PREPDT        Prepare to compare data
542 1CBF9 78F0         GOSUB  VFYCRD        Verify
543 1CBFD 49B          GOC    VFY000
544 1CC00 831          ?XM=0
545 1CC03 50           GOYES  VFY060
546 1CC05 5FD          GONC   VFY035        B.E.T.
547 1CC08 862  VFY060  ?ST=0  2             Last track?
548 1CC0B F2           GOYES  VFY080        No.
549 1CC0D 119          C=R1
550 1CC10 70F5         GOSUB  csrw5         Yes.  This # 0-bytes to expect.
551 1CC14 81E          CSRB
552 1CC17 81E          CSRB                 Will read # mod 4 bytes.
553 1CC1A D7           D=C    A             # reads to do.
554 1CC1C CF   VFY070  D=D-1  A             Verify zero-fill loop
555 1CC1E 4B1          GOC    VFY080        Go if done verifying.
556 1CC21 851          ST=1   1
557 1CC24 77B4         GOSUB  RD8SV         Read.
558 1CC28 4E8          GOC    VFY000        Go if error.
559 1CC2B 861          ?ST=0  1             Data error?
560 1CC2E 7B           GOYES  VFY035        Yes...Verify error.
561 1CC30 27           P=     7
```

```
562 1CC32 91C          ?R#0   WP             Zero?
563 1CC35 0B           GOYES  VFY035         No,  verify error.
564 1CC37 54E          GONC   VFY070         Yes.  On to next read.
565 1CC3A 7574  VFY080 GOSUB  CRDOFF         Turn off card reader.
566 1CC3E 7A63         GOSUB  TRKDON
567 1CC42 7271         GOSUB  RTODP
568 1CC46 175          D1=D1+ 2*3            Point at trksize
569 1CC49 D2           C=0    A
570 1CC4B 15F3         C=DAT1 4
571 1CC4F C6           C=C+C  A              Tracksize in nibbles
572 1CC51 111          A=R1
573 1CC54 CA           A=A+C  A
574 1CC56 101          R1=A                  Update file pointer
575 1CC59 1C3          D1=D1- 2*2
576 1CC5C 14B          A=DAT1 B
577 1CC5F B64          A=A+1  B              Increment track#
578 1CC62 149          DAT1=A B
579 1CC65 670E         GOTO   FILC40         Write next track
580             ********************************************************************
581             ********************************************************************
582             **
583             ** Name:(S) pWCRD8  -  Poll To Write Copycode 8 File To Card
584             **
585             ** Category:   POLL
586             **
587             ** Type:       POLL
588             **
589             ** Purpose:
590             **      Allow handler to copy a file with copycode of 8 out to
591             **      card.
592             **
593             ** Should poll be "Handled" (return with XM=0)?:
594             **      Yes, if you do the copy.
595             **
596             ** Meaning of "Handling" Poll (what does code do if handled?):
597             **      The copy has been performed.  The WHOLE thing...
598             **      prompting, writing, verifying, etc.  The copy code
599             **      will perform a normal exit.  If poll is not handled,
600             **      copy code performs an eror exit.
601             **
602             ** Entry conditions for handler (registers, ST, RAM, etc.):
603             **      Carry set on entry.
604             **      B[A] = Poll number.
605             **      HEX mode.
606             **      P=0.
607             **      Card header buffer (ID=bCARD) has been allocated and
608             **      set up (as per FILCRD header) with:
609             **        Name
610             **        Filetype
611             **        Creation date
612             **        Subformat and track#.
613             **      R1[A] points at start of file header.
614             **      R2[A] points at card header I/Obuffer (past header).
615             **      A[3-0] contains filetype.
616             **
```

```
617             ** Normal exit conditions from handler if handled (ST, RAM,
618             ** registers, etc.):
619             **      HEX mode.
620             **      XM=0.
621             **
622             ** Normal exit conditions from handler if not handled (ST, RAM,
623             ** registers, etc.):
624             **      HEX mode.
625             **      XM=1.
626             **
627             ** Available subroutine levels:
628             **      5
629             **
630             ** What registers/RAM may be used if handled?:
631             **      A-D, D0, D1, P, R0-R4, all scratch RAM.
632             **
633             ** What registers/RAM may be used if not handled?:
634             **      A-D, D0, D1, P, R0-R4, all scratch RAM.
635             **
636             ** Envisioned application(s):
637             **      Somebody's got to know how to copy out a file with a
638             **      crazy copycode like 8.
639             **
640             ** History:
641             **
642             **    Date     Programmer             Modification
643             **  --------   ----------   ----------------------------------
644             **  08/01/83   NM           Added documentation
645             **
646             ********************************************************************
647             ********************************************************************
```

```
                        STITLE Copy to card utilities
                ****************************************************************
                ****************************************************************
                **
                ** Name:    MAXTRK  -  Compute Max Trk# For Card Set
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Compute # tracks a card set will require.
                **
                ** Entry:
                **      R2 points at card header I/Obuffer.
                **
                ** Exit:
                **      A[B] = max trk#.
                **      D1 points at filesize field in header I/Obuffer.
                **
                ** Calls:      LCTRKS, IDIVA.
                **
                ** Uses.......
                **              A,B,C,D1,P.
                **
                ** Stk lvls:   1
                **
                ** Detail:
                **      Reads filesize from card header I/Obuffer.
                **
                ** Algorithm:
                **      Read filesize.
                **      Compute filesize/tracksize.
                **      Add 1 to result if remainder > 0.
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **   --------   ----------   ----------------------------------
                **   06/21/82   NM           Added documentation
                **
                ****************************************************************
                ****************************************************************
1CC69 11A    MAXTRK C=R2
1CC6C 135           D1=C
1CC6F 179           D1=D1+ 2*5              Point at filesize
1CC72 D0            A=0    A
1CC74 15B3          A=DAT1 4                #bytes in this file
1CC78 7131          GOSUB  LCTRKS           #bytes/track
1CC7C 72D1          GOSUB  idiva            Filesize/cardsize=#tracks
1CC80 8A9           ?B=0   A                Any remainder?
1CC83 00            RTNYES                  No
1CC85 B64           A=A+1  B                Yes, add 1 trk for frac card
1CC88 01            RTN
                ****************************************************************
                ****************************************************************
                **
```

```
                ** Name:    PREPHD  -  Prepare To Write/verify Hdr Field
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Set up pointers to do a write or verify of card header
                **      field.
                **
                ** Entry:
                **      R2 points at card header I/Obuffer.
                **
                ** Exit:
                **      D0=R2.
                **      C[A]=size of header I/Obuffer (72 nibbles).
                **      P=0.
                **
                ** Calls:      None.
                **
                ** Uses.......
                **              D0,P,C.
                **
                ** Stk lvls:   0
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------   ----------   ------------------------------------
                **  06/21/82   NM           Added documentation
                **
                ********************************************************************
                ********************************************************************
1CC8A 11A    PREPHD C=R2
1CC8D 134           D0=C                    Point at header
1CC90 D2            C=0    A
1CC92 20            P=     0
1CC94 3184          LC(2)  72               Will do 72 nibs
1CC98 01            RTN
                ********************************************************************
                ********************************************************************
                **
                ** Name:    PREPDT  -  Prepare For Write Or Verify
                **
                ** Category:   LOCAL
                **
                ** Purpose:
                **      Set up pointers for a write or a verify.
                **
                ** Entry:
                **      R2 points at header I/Obuffer.
                **      R1[A] points at start of data for this track.
                **      R1[9-5]=size of padding at end of card set write.
                **      S2=1 if this is last card in set.
                **
                ** Exit:
                **      D0=R1.
```

```
758            **      D1=R2.
759            **      C[A]=tracksize in nibbles.
760            **
761            ** Calls:      RTODP.
762            **
763            ** Uses.......
764            **            A,C,D0,D1.
765            **
766            ** Stk lvls:   1
767            **
768            ** Detail:
769            **      Reads tracksize from header I/Obuffer.
770            **
771            ** History:
772            **
773            **    Date     Programmer            Modification
774            **  --------   ----------   ---------------------------------
775            **  06/21/82   NM           Added documentation
776            **
777            ********************************************************************
778            ********************************************************************
779 1CC9A 7A11  PREPDT GOSUB  RTODP
780 1CC9E 175         D1=D1+ 2*3
781 1CCA1 D2          C=0    A
782 1CCA3 15F3        C=DAT1 4              Tracksize in bytes
783 1CCA7 862         ?ST=0  2              Is this last track?
784 1CCAA B0          GOYES  PREPD1         No.
785 1CCAC 111         A=R1                  Yes.
786 1CCAF 7F55        GOSUB  asrw5          Padding size in bytes.
787 1CCB3 E2          C=C-A  A              Subtr padding size from trksize
788 1CCB5 C6   PREPD1 C=C+C  A              Size of data field in nibbles.
789 1CCB7 01          RTN
790            ********************************************************************
791            ********************************************************************
792            **
793            ** Name:    TOCARD  -  Write To Card
794            **
795            ** Category:   LOCAL
796            **
797            ** Purpose:
798            **      Write specified amount of memory to card.
799            **
800            ** Entry:
801            **      Card is moving.
802            **      BRKSRV has been set if needed.
803            **      D0 points at memory to be written.
804            **      C[A]=#nibs to write.
805            **
806            ** Exit:
807            **      Carry clear if write successful, else WRIT8S reported
808            **        an error.
809            **
810            ** Calls:      WRIT8S.
811            **
812            ** Uses.......
```

```
                **            A,C,D,P,D1,B[B],S1
                **            {A,B,C,D,D0,R0,scratch RAM if error}
                **
                ** Stk lvls:   2
                **
                ** Detail:
                **      If there is a partial write at the end (<8 nibs), it
                **      is padded with zeroes.  Otherwise, there are no
                **      trailing zeroes.
                **
                ** Algorithm:
                **      D[A]=C[A] div 8; D[S]=[C[A] mod 8] * 2.
                **      { # full writes to D[A], size of partial write * 2 to
                **        D[S] }
                **   1: If D[A]=0 then goto 2.
                **      Read data at D0.
                **      Write data to FIFO.
                **      RTNC if error.
                **      Goto 1.
                **   2: P=D[S]/2 - 1.
                **      If carry set on subtract { D[S] was 0 } then RTNCC.
                **      A=0.
                **      Read data[WP] at D0.
                **      Write data to FIFO.
                **      RTN.
                **
                ** History:
                **
                **    Date      Programmer            Modification
                **  --------   ----------   ----------------------------------
                **  06/21/82   NM           Added documentation
                **
                ********************************************************************
                ********************************************************************
1CCB9 AD3    TOCARD D=0    M
1CCBC D7            D=C    A
1CCBE A77           D=D+D  W
1CCC1 817           DSRC                  #writes in d[a],frc*2 in [s]
1CCC4 CF     TOCA10 D=D-1  A              Any more full writes?
1CCC6 421           GOC    TOCA20         No
1CCC9 15A7          A=DAT0 8              Yes, read data
1CCCD 167           D0=D0+ 8
1CCD0 7514          GOSUB  WRIT8S         Write it
1CCD4 5FE           GONC   TOCA10         Go if successful
1CCD7 01            RTN                   Return Carry set
1CCD9 ACB    TOCA20 C=D    S
1CCDC 81E           CSRB
1CCDF AF0           A=0    W              Room for frac write
1CCE2 80DF          P=C    15             Size of final write in nibs
1CCE6 0D            P=P-1                 Anything to write?
1CCE8 4A0           GOC    TOCA30         No.
1CCEB 1521          A=DAT0 WP             Read frac write
1CCEF 69F3          GOTO   WRIT8S         Write
1CCF3 03     TOCA30 RTNCC
                ********************************************************************
```

```
868             ****************************************************************
869             **
870             ** Name:    VFYCRD  -  Compare Card To Memory
871             **
872             ** Category:   LOCAL
873             **
874             ** Purpose:
875             **      Verify that card was written correctly.
876             **
877             ** Entry:
878             **      D0 points at memory.
879             **      C[A]=#nibs to compare.
880             **
881             ** Exit:
882             **      Carry set if READ8S reported error.
883             **      Else XM set if verify error.
884             **
885             ** Calls:      RD8SV.
886             **
887             ** Uses.......
888             **              A,C,D,P,D1,B[B],XM,S1
889             **              {A,B,C,D,D0,R0,scratch RAM if error}
890             **
891             ** Stk lvls:   2
892             **
893             ** Algorithm:
894             **      XM=0
895             **      D[A]=C[A] div 8; D[S]=[C[A] mod 8] * 2.
896             **      { # full writes to D[A], size of partial write * 2 to
897             **        D[S] }
898             **    1: If D[A]=0 then goto 2.
899             **      Read data at D0.
900             **      Read data from FIFO.
901             **      RTNC if error.
902             **      RTNSXM (with CC) if compare failure.
903             **      Goto 1.
904             **    2: P=D[S]/2 - 1.
905             **      If carry set on subtract { D[S] was 0 } then RTNCC.
906             **      A=0.
907             **      Read data[WP] at D0.
908             **      Read data from FIFO.
909             **      RTNC if error.
910             **      RTNSXM (with CC) if compare failure.
911             **      RTNCC.
912             **
913             ** History:
914             **
915             **    Date      Programmer              Modification
916             **  --------    ----------   ---------------------------------
917             **  06/21/82    NM           Added documentation
918             **
919             ****************************************************************
920             ****************************************************************
921 1CCF5 821    VFYCRD XM=0
922 1CCF8 AD3          D=0    M
```

```
 923 1CCFB D7           D=C    A
 924 1CCFD A77          D=D+D  W
 925 1CD00 817          DSRC                  #writes in d[a], frac*2 in [s]
 926 1CD03 CF    VFYC10 D=D-1  A              Any more full-read compares?
 927 1CD05 412          GOC    VFYC20         No
 928 1CD08 851          ST=1   1              Tell WAIT that this is VFY
 929 1CD0B 70D3         GOSUB  RD8SV          Read
 930 1CD0F 400          RTNC                  Return if error or abort
 931 1CD12 861          ?ST=0  1              Data error found?
 932 1CD15 F3           GOYES  VFYC30         Yes.  Verify error.
 933 1CD17 15E7         C=DAT0 8              Data to compare
 934 1CD1B 167          DO=DO+ 8
 935 1CD1E 27           P=     7
 936 1CD20 912          ?A=C   WP             Match?
 937 1CD23 0E           GOYES  VFYC10         Yes... next compare
 938 1CD25 00           RTNSXM                No.  verify failure.
 939 1CD27 81F   VFYC20 DSRB
 940 1CD2A A4F          D=D-1  S              Frac read to compare?
 941 1CD2D 45C          GOC    TOCA30         No.  done
 942 1CD30 851          ST=1   1              Tell WAIT that this is VFY
 943 1CD33 78A3         GOSUB  RD8SV          Yes.  last read.
 944 1CD37 400          RTNC
 945 1CD3A 861          ?ST=0  1              Data error found?
 946 1CD3D 71           GOYES  VFYC30         Yes.  Verify error.
 947 1CD3F ACB          C=D    S
 948 1CD42 80DF         P=C    15             Size of last compare
 949 1CD46 AF2          C=0    W
 950 1CD49 1561         C=DAT0 WP             Read data from memory
 951 1CD4D 27           P=     7              Compare entire read (w/0's)
 952 1CD4F 912          ?A=C   WP             Match?
 953 1CD52 1A           GOYES  TOCA30         Yes.  rtncc
 954 1CD54 21    VFYC30 P=     1
 955 1CD56 0D           P=P-1                 Clear carry.
 956 1CD58 00           RTNSXM                RTN xm and cc.
 957             ********************************************************************
 958             ********************************************************************
 959             **
 960             ** Name:    IMPFLD  -  Compute Implementation Field
 961             **
 962             ** Category:   LOCAL
 963             **
 964             ** Purpose:
 965             **      Compute implementation field for output to card.
 966             **
 967             ** Entry:
 968             **      DO points at file header I/Obuffer.
 969             **
 970             ** Exit:
 971             **      RTNNC if copy code=8.
 972             **      Else C[7-0] contains implementation field.
 973             **
 974             ** Calls:      None.
 975             **
 976             ** Uses.......
 977             **             DO,C.
```

```
 978                    **
 979                    ** Stk lvls:   0
 980                    **
 981                    ** Detail:
 982                    **      Copy code     Implementation Field
 983                    **      ---------     --------------------
 984                    **          0         File length in nibbles.
 985                    **          1         Implementation field from file
 986                    **                    (immediately follows chain length field).
 987                    **          2         File length/16.
 988                    **          4         00000000.
 989                    **          8         (Don't know).
 990                    **
 991                    ** History:
 992                    **
 993                    **    Date      Programmer            Modification
 994                    **  --------   ----------   ---------------------------------
 995                    **  06/21/82   NM           Added documentation
 996                    **
 997                    ********************************************************************
 998                    ********************************************************************
 999 1CD5A 16F   IMPFLD D0=D0+ 16
1000 1CD5D 164         D0=D0+ 5              Point at flags
1001 1CD60 AF2         C=0    W
1002 1CD63 1564        C=DATO S              Read copy code
1003 1CD67 16A         D0=D0+ 11             Point at file chain length
1004 1CD6A A46         C=C+C  S              Copy code=8?
1005 1CD6D 458         GOC    TOCA30         Yes.  rtncc.
1006 1CD70 A46         C=C+C  S              Copy code=4?
1007 1CD73 400         RTNC                  Yes.  imp field = zeroes
1008 1CD76 146         C=DATO A              Read link field
1009 1CD79 136         CDOEX
1010 1CD7C 184         D0=D0- 5              Subtract length of link field
1011 1CD7F 136         CDOEX                 File length
1012 1CD82 A46         C=C+C  S              Copy code=2?
1013 1CD85 560         GONC   IMPF10         No
1014 1CD88 F6          CSR    A              Yes.  divide len by 16
1015 1CD8A 01          RTN
1016 1CD8C 94A  IMPF10 ?C=0   S              Copy code=1?
1017 1CD8F 00          RTNYES                No.  return file len in nibs
1018 1CD91 164         D0=D0+ 5              Yes.  point at imp field
1019 1CD94 15E7        C=DATO 8              Return imp field
1020 1CD98 02          RTNSC
1021                   ********************************************************************
1022                   ********************************************************************
1023                   **
1024                   ** Name:    FNDPRT  -  Fill In Partial Card Recovery Fields
1025                   **
1026                   ** Category:   LOCAL
1027                   **
1028                   ** Purpose:
1029                   **      Fill in partial card recovery fields in card header.
1030                   **
1031                   ** Entry:
1032                   **      R2 points at card header.
```

```
1033            **
1034            ** Exit:
1035            **      Carry clear if successful.
1036            **
1037            ** Calls:       D1+27B.
1038            **
1039            ** Uses.......
1040            **              C,D1.
1041            **
1042            ** Stk lvls:    1
1043            **
1044            ** Detail:
1045            **      Since partial card recovery was designed but is not
1046            **      being used, this rather simple routine simply fills
1047            **      the appropriate fields with zeroes.  In case partial
1048            **      card recovery is ever used, this is what goes into
1049            **      the partial card recovery fields:
1050            **
1051            **      Partial statement status (pss) field and s1 and s2
1052            **      contain information for recovery of files which were
1053            **      only partially read:
1054            **
1055            **          partial statement status (1):
1056            **            bit#    meaning
1057            **            ----    -------
1058            **              0     file unrecoverable(0)/recoverable(1) if
1059            **                    not all cards read
1060            **              1     fill(0)/pack(1) out missing tracks
1061            **              2     variable(0)/fixed(1) length records
1062            **              3     zero(0)/one(1)-fill missing tracks
1063            **                    (if bit #1 = 0)
1064            **              4
1065            **              5
1066            **              6     add 1 nib to s2
1067            **              7     add 1 nib to s1
1068            **      For files with bit 0 set and bit 2 clear, s1 and s2
1069            **      have the following meaning:
1070            **          s1: size of first partial line (start of data to
1071            **              to end of line): in bytes
1072            **          s2: size of last partial line (start of line to
1073            **              end of data): in bytes
1074            **      For files with bit 0 set and bit 2 set:
1075            **          s1: record size
1076            **              (this is needed for file header and for
1077            **               packing/zeroing missing records)
1078            **
1079            **
1080            ** Algorithm:
1081            **      Zero-fill partial card recovery fields.
1082            **
1083            ** History:
1084            **
1085            **    Date      Programmer            Modification
1086            **  --------    ----------   ------------------------------------
1087            **  06/21/82    NM           Added documentation
```

```
1088                **
1089                **********************************************************************
1090                **********************************************************************
1091 1CD9A 11A    FNDPRT C=R2
1092 1CD9D 135          D1=C                    Point at card header buffer
1093 1CDA0 7B31         GOSUB  D1+27B           Point at PSS
1094 1CDA4 AF2          C=0    W
1095 1CDA7 15D9         DAT1=C 2*5             Zero out pss area
1096 1CDAB 03           RTNCC
1097                **********************************************************************
1098                **********************************************************************
1099                **
1100                ** Name:    LCTRKS  -  Load A Constant
1101                **
1102                ** Category:   LOCAL
1103                **
1104                ** Purpose:
1105                **      Load constant corresponding to trksize in bytes.
1106                **
1107                ** Entry:
1108                **
1109                ** Exit:
1110                **      P=0.
1111                **      C[A]=Tracksize in bytes.
1112                **
1113                ** Calls:      None.
1114                **
1115                ** Uses.......
1116                **              P,C[A].
1117                **
1118                ** Stk lvls:   0
1119                **
1120                ** History:
1121                **
1122                **    Date      Programmer              Modification
1123                **   --------   ----------   ----------------------------------
1124                **   10/06/82   NM           Proudly authored.
1125                **
1126                **********************************************************************
1127                **********************************************************************
1128 1CDAD 20     LCTRKS P=     0
1129 1CDAF 34A8         LC(5)  TRKSIZ
           200
1130 1CDB6 01           RTN
```

```
1131                    STITLE Card reader utilities
1132                ****************************************************************
1133                ****************************************************************
1134                **
1135                ** Name:    RTODP   -  Scratch Regs To DPs
1136                ** Name:    R1TODO  -  Scratch Regs To DPs
1137                **
1138                ** Category:   LOCAL
1139                **
1140                ** Purpose:
1141                **      Copy contents of scratch registers to DPs.
1142                **
1143                ** Entry:
1144                **
1145                ** Exit:
1146                **
1147                ** Calls:      None.
1148                **
1149                ** Uses.......
1150                **             A,D0,D1.
1151                **
1152                ** Stk lvls:   0
1153                **
1154                ** Algorithm:
1155                **   RTODP: D1=R2; D0=R1.
1156                **   R1TODO: D0=R1.
1157                **
1158                ** History:
1159                **
1160                **    Date     Programmer              Modification
1161                **  --------   ----------   ----------------------------------
1162                **  10/06/82   NM           Wrote.
1163                **
1164                ****************************************************************
1165                ****************************************************************
1166 1CDB8 112    RTODP  A=R2
1167 1CDBB 131           D1=A
1168 1CDBE 111    R1TODO A=R1
1169 1CDC1 130           D0=A
1170 1CDC4 01            RTN
1171                ****************************************************************
1172                ****************************************************************
1173                **
1174                ** Name:    GETSOC  -  Read First Part Of SOC Field
1175                **
1176                ** Category:   LOCAL
1177                **
1178                ** Purpose:
1179                **      Read first subfield from SOC field.
1180                **
1181                ** Entry:
1182                **      BRKSRV has not been set yet.
1183                **      HEX mode.
1184                **
1185                ** Exit:
```

```
1186                **      Results of read in A[7-0].
1187                **      Expected result in C[7-0] ("HPCV").
1188                **      P=7 for comparison.
1189                **      CARRY as set by READ8.
1190                **
1191                ** Calls:      READFL, READ8.
1192                **
1193                ** Uses.......
1194                **              A[7-0],C[7-0],P,D1,B[B],S1
1195                **              {A,B,C,D,D0,R0,scratch RAM if error}
1196                **
1197                ** Stk lvls:   2
1198                **
1199                ** History:
1200                **
1201                **    Date      Programmer              Modification
1202                **  --------    ----------   ----------------------------------
1203                **  06/21/82    NM           Added documentation
1204                **
1205                ********************************************************************
1206                ********************************************************************
1207 1CDC6           GETSOC
1208 1CDC6 70F2            GOSUB   READFL          Set flgsrv
1209 1CDCA 7103            GOSUB   READ8           Read soc field
1210 1CDCE 20              P=      0
1211 1CDD0 3784            LCASC   \VCPH\
           0534
           65
1212 1CDDA 27              P=      7
1213 1CDDC 01              RTN
1214                ********************************************************************
1215                ********************************************************************
1216                **
1217                ** Name:    GETSIZ  -  Read Size Subfield From SOC
1218                **
1219                ** Category:   LOCAL
1220                **
1221                ** Purpose:
1222                **      Read size subfield and prepare for testing it.
1223                **
1224                ** Entry:
1225                **      Card is moving.  We have just read "HPCV" subfield
1226                **        from header.
1227                **
1228                ** Exit:
1229                **      Result of read in A[3-0].
1230                **      Expected result in C[3-0].
1231                **      P=3 for comparison.
1232                **      Carry as set by READ8S.
1233                **
1234                ** Calls:      READ8S.
1235                **
1236                ** Uses.......
1237                **              A[7-0],C[5-0],P,D1,B[B],S1
1238                **              {A,B,C,D,D0,R0,scratch RAM if error}
```

```
1239                ** 
1240                ** Stk lvls:   2
1241                ** 
1242                ** History:
1243                ** 
1244                **    Date     Programmer          Modification
1245                **  --------   ----------  ---------------------------------
1246                **  06/21/82   NM          Added documentation
1247                ** 
1248                ********************************************************************
1249                ********************************************************************
1250 1CDDE 7AF2  GETSIZ GOSUB  READ8S         Read size from soc field
1251 1CDE2 20           P=     0
1252 1CDE4 33CB         LC(4)  (TRKSIZ)+50    We expect this trksize
          20
1253 1CDEA 23           P=     3
1254 1CDEC 01           RTN
1255                ********************************************************************
1256                ********************************************************************
1257                ** 
1258                ** Name:    CHKSUM  -  Compute 2-byte Checksum
1259                ** 
1260                ** Category:   LOCAL
1261                ** 
1262                ** Purpose:
1263                **      Compute 2-byte checksum of a block of memory.
1264                ** 
1265                ** Entry:
1266                **      D0 points at start of memory to be checksummed.
1267                **      C[A]=#nibbles to be checksummed.
1268                ** 
1269                ** Exit:
1270                **      2-byte checksum in A[3-0].
1271                **      P=3.
1272                ** 
1273                ** Calls:      CHKS70.
1274                ** 
1275                ** Uses.......
1276                **             A,B,C,P,D0.
1277                ** 
1278                ** Stk lvls:   1
1279                ** 
1280                ** Detail:
1281                **      Uses standard checksum with wraparound carry.
1282                ** 
1283                ** Algorithm:
1284                **      Clear B for checksum.
1285                **      C[3-0]=#nibs div 16; C[S]=#nibs mod 16.
1286                **    1: If C[3-0]=0 then goto 2.
1287                **      Read 16 nibbles into A at D0.
1288                **      Increment D0 by 16.
1289                **      Add A to B with wraparound carry.
1290                **      Goto 1.
1291                **    2: A=0.
1292                **      Read partial word (C[S] nibbles) into A at D0.
```

```
1293                **      Add A to B with wraparound carry.
1294                **      B[7-0]=B[7-0]+B[15-8] with wraparound carry.
1295                **      A[3-0]=B[3-0]+B[7-4] with wraparound carry.
1296                **      RTN.
1297                **
1298                ** History:
1299                **
1300                **    Date      Programmer              Modification
1301                **  --------    ----------   ------------------------------------
1302                **  06/21/82    NM           Added documentation
1303                **
1304                ********************************************************************
1305                ********************************************************************
1306 1CDEE AF1    CHKSUM B=0    W               Room for checksum
1307 1CDF1 80D0         P=C     0               Frac word in P
1308 1CDF5 F6           CSR     A               # Words in C[A]
1309 1CDF7 CE     CHKS10 C=C-1   A               Any more full words?
1310 1CDF9 451          GOC     CHKS20          No
1311 1CDFC 1527         A=DAT0  W               Yes, read
1312 1CE00 16F          DO=DO+  16
1313 1CE03 A78          B=B+A   W               Add... carry?
1314 1CE06 50F          GONC    CHKS10          No
1315 1CE09 B75          B=B+1   W               Yes, wraparound
1316 1CE0C 5AE          GONC    CHKS10          B.E.T.
1317 1CE0F 0D     CHKS20 P=P-1                   Frac word=0?
1318 1CE11 421          GOC     CHKS30          Yes
1319 1CE14 AF0          A=0     W               No, create space
1320 1CE17 1521         A=DAT0  WP              Read partial word
1321 1CE1B A78          B=B+A   W               Add... carry?
1322 1CE1E 550          GONC    CHKS30          No
1323 1CE21 B75          B=B+1   W               Yes, wraparound
1324 1CE24 AF4    CHKS30 A=B    W
1325 1CE27 27           P=      7
1326 1CE29 BF4    CHKS40 ASR    W               Loop to align half-words
1327 1CE2C 0D           P=P-1
1328 1CE2E 5AF          GONC    CHKS40
1329 1CE31 27           P=      7
1330 1CE33 7010         GOSUB   CHKS70          Compress to 8 nibbles
1331 1CE37 A98          B=A     WP
1332 1CE3A B94          ASR     WP
1333 1CE3D B94          ASR     WP
1334 1CE40 B94          ASR     WP
1335 1CE43 F4           ASR     A               Align quarter-words
1336 1CE45 23           P=      3
1337 1CE47 A10    CHKS70 A=A+B   WP              Compress to 4 nibbles
1338 1CE4A 500          RTNNC
1339 1CE4D B14          A=A+1   WP              Wraparound carry
1340 1CE50 01           RTN
1341 1CE52 8D00 idiva   GOVLNG =IDIVA
          000
1342                *
1343                ********************************************************************
1344                ********************************************************************
1345                **
1346                ** Name:    STDRG?  -  Check If File In Standard Range
```

```
1347                ** 
1348                ** Category:   FILUTL
1349                ** 
1350                ** Purpose:
1351                **      Check filetype to determine if it is in standard range.
1352                ** 
1353                ** Entry:
1354                **      Filetype in A[A].
1355                ** 
1356                ** Exit:
1357                **      Carry set iff file is NOT in standard range.
1358                **      P=0.
1359                ** 
1360                ** Calls:      None.
1361                ** 
1362                ** Uses.......
1363                **             C[A],P.
1364                ** 
1365                ** Stk lvls:   0
1366                ** 
1367                ** History:
1368                ** 
1369                **    Date      Programmer              Modification
1370                **  --------   ----------   ----------------------------------
1371                **  06/25/82   NM           Added documentation
1372                ** 
1373                ********************************************************************
1374                ********************************************************************
1375 1CE59 20     =STDRG? P=     0
1376 1CE5B 345D         LCHEX  0E0D5          Lowbound of std range
           0E0
1377 1CE62 8B2          ?A<C   A
1378 1CE65 00           RTNYES                Return if below
1379 1CE67 32FF         LCHEX  3FF            Highbound of std range
           3
1380 1CE6C E2           C=C-A  A              Set carry if a>c
1381 1CE6E 01           RTN
1382                ********************************************************************
1383                ********************************************************************
1384                ** 
1385                ** Name:    RDSOC   -  Get Past SOC And WPROT Field
1386                ** 
1387                ** Category:   LOCAL
1388                ** 
1389                ** Purpose:
1390                **      Check SOC field and read WPROT field.
1391                ** 
1392                ** Entry:
1393                **      None.
1394                ** 
1395                ** Exit:
1396                **      Carry set if read failure (reported before return).
1397                **      Else A[7-0]=first 4 bytes of WPROT field.
1398                ** 
1399                ** Calls:      GETSOC, GETSIZ, READFL, READ8S {falls through to
```

```
1400                ** UNKCD if unknown card}.
1401                **
1402                ** Uses.......
1403                **          C[5-0],P,D1,B[B],S1
1404                **          {A,B,C,D,D0,R0,scratch RAM if error}
1405                **
1406                ** Stk lvls:  3
1407                **
1408                ** Algorithm:
1409                **      GETSOC; RTNSC if error.
1410                **      Compare to expected result {"HPCV"}, warn with eNCVD
1411                **        and RTNSC if not match.
1412                **      GETSIZ; RTNSC if error.
1413                **      Compare to expected result; warn with eNCVD and RTNSC
1414                **        if not match.
1415                **      Set FLGSRV.
1416                **      Read first 4 bytes of WPROT field.
1417                **      RTN.
1418                **
1419                ** History:
1420                **
1421                **    Date      Programmer          Modification
1422                **  --------   ----------  ----------------------------------
1423                **  06/25/82   NM          Added documentation
1424                **
1425                ********************************************************************
1426                ********************************************************************
1427 1CE70 725F  RDSOC  GOSUB  GETSOC        Get soc field
1428 1CE74 400         RTNC                 Return if error
1429 1CE77 912         ?A=C   WP            Is this HPCV card?
1430 1CE7A 60          GOYES  RDS050        Yes
1431 1CE7C 62C0  RDS049 GOTO   UNKCD        No
1432 1CE80 7A5F  RDS050 GOSUB  GETSIZ        Read card size
1433 1CE84 400         RTNC
1434 1CE87 916         ?A#C   WP            Right size?
1435 1CE8A 2F          GOYES  RDS049
1436 1CE8C 7A22        GOSUB  READFL        Set flgsrv
1437 1CE90 6B42        GOTO   READ8S        Read first 8 nibs of wprot
1438                ********************************************************************
1439                ********************************************************************
1440                **
1441                ** Name:    CR??    -  Prepare For Card Reader Operations
1442                **
1443                ** Category:  LOCAL
1444                **
1445                ** Purpose:
1446                **      Verify presence of hardware; clear display.
1447                **
1448                ** Entry:
1449                **      None.
1450                **
1451                ** Exit:
1452                **      Through MFERR if card reader hardware not present.
1453                **      Else RTN.
1454                **      HEX mode.
```

```
1455                **
1456                ** Calls:      CRPRS?, crlfnd {falls through}.
1457                **
1458                ** Uses.......
1459                **             A,B,C,D,P,D0,D1,R0
1460                **
1461                ** Stk lvls:   3
1462                **
1463                ** Algorithm:
1464                **      SETHEX.
1465                **      Read (=CR)+14.  If zero {CR not present} error out.
1466                **      Send COCRLF to display.
1467                **
1468                ** History:
1469                **
1470                **    Date     Programmer              Modification
1471                ** --------   ----------   ----------------------------------
1472                ** 06/25/82   NM           Added documentation
1473                **
1474                ******************************************************************
1475                ******************************************************************
1476 1CE94 04     CR??   SETHEX
1477 1CE96 7F00           GOSUB  CRPRS?
1478 1CE9A 5A0            GONC   CR??20
1479 1CE9D 1D00   CR??10 D1=(2) =eDVCNF
1480 1CEA1 6791           GOTO   errout
1481 1CEA5 6820   CR??20 GOTO   crlfnd
1482                *-
1483                *-
1484 1CEA9 1F41   CRPRS? D1=(5) (=CR)+#14
           0C2
1485               TEST   IF     TEST
1489               TEST   ENDIF
1490 1CEB0 1574           C=DAT1 S
1491 1CEB4 A4E            C=C-1  S
1492 1CEB7 01             RTN
1493                ******************************************************************
1494                ******************************************************************
1495                **
1496                ** Name:    RTNABT  -  Abort Return From Card Reader
1497                **
1498                ** Category:   LOCAL
1499                **
1500                ** Purpose:
1501                **      Perform ABORT from card reader operations.
1502                **
1503                ** Entry:
1504                **      Code has decided to abort.
1505                **
1506                ** Exit:
1507                **      NXTSTM.
1508                **
1509                ** History:
1510                **
1511                **     Date     Programmer              Modification
```

```
1512            **   --------   ----------   ---------------------------------
1513            **    05/28/82   NM           Wrote, etc.
1514            **
1515            ********************************************************************
1516            ********************************************************************
1517            *!!
1518            *!! Replace following five lines with:
1519            *!!
1520            *!! RTNABT GOSUB  FINCRD         Clear display & delete buffer
1521            *!!        GOVLNG =INPO10        Psuedo-error exit
1522            *!! RTNAB+ GOSUB  FINCRD         Clear display & delete buffer
1523            *!! NEXTST GOLONG =REN180        To NXTSTM
1524            *!!
1525 1CEB9 85E   RTNABT ST=1   14              Set Don't-continue flag
1526 1CEBC 7E00  RTNAB+ GOSUB  crlfnd
1527 1CEC0 7110        GOSUB  noscrl
1528 1CEC4 7B30  NEXTST GOSUB  BUFDAL
1529 1CEC8 8C00        GOLONG =REN180        GOTO NXTSTM
          00
1530            *
1531 1CECE 8D00 =crlfnd GOVLNG =CRLFND
          000
1532 1CED5 8D00  noscrl GOVLNG =NOSCRL
          000
1533            *
1534            ********************************************************************
1535            ********************************************************************
1536            **
1537            ** Name:    D1+xxB  -  Add Xx Bytes To D1
1538            **
1539            ** Category:   LOCAL
1540            **
1541            ** Purpose:
1542            **      Increment D1 by 13, 21, 27 or 29 bytes.
1543            **
1544            ** Entry:
1545            **
1546            ** Exit:
1547            **
1548            ** Calls:      None.
1549            **
1550            ** Uses.......
1551            **             D1.
1552            **
1553            ** Stk lvls:   0
1554            **
1555            ** History:
1556            **
1557            **    Date     Programmer             Modification
1558            **  --------   ----------   ---------------------------------
1559            **  10/07/82   NM           Authored.
1560            **
1561            ********************************************************************
1562            ********************************************************************
1563 1CEDC 173   D1+29B D1=D1+ 2*2
```

```
1564 1CEDF 17B   D1+27B D1=D1+ 2*6
1565 1CEE2 17F   D1+21B D1=D1+ 2*8
1566 1CEE5 179   D1+13B D1=D1+ 2*5
1567 1CEE8 17F          D1=D1+ 2*8
1568 1CEEB 01           RTN
1569            ****************************************************************
1570            ****************************************************************
1571            **
1572            ** Name:    IOAL36  -  Allocate CR I/O Buffer
1573            ** Name:    IOAL+   -  Allocate CR I/O Buffer
1574            **
1575            ** Category:   LOCAL
1576            **
1577            ** Purpose:
1578            **      Allocate card reader I/O buffer.
1579            **
1580            ** Entry:
1581            **   IOAL36: None.
1582            **   IOAL+: P=0, buffersize in C[A] (nibs 3-4 = 0)
1583            **
1584            ** Exit:
1585            **      P=0.
1586            **      Carry set iff buffer successfully allocated.
1587            **
1588            ** Calls:      I/OAL+ (falls through).
1589            **
1590            ** Uses.......
1591            **             A,B,C,D,D0,D1,P.
1592            **
1593            ** Stk lvls:   3
1594            **
1595            ** History:
1596            **
1597            **    Date     Programmer              Modification
1598            **  --------   ----------   ----------------------------------
1599            **  10/08/82   NM           Wrote.
1600            **
1601            ****************************************************************
1602            ****************************************************************
1603 1CEED 20    IOAL36 P=     0
1604 1CEEF D2           C=0    A
1605 1CEF1 3184         LC(2)  2*36
1606 1CEF5 D5    IOAL+  B=C    A
1607 1CEF7 3270         LC(3)  =bCARD
           8
1608 1CEFC 8D00         GOVLNG =I/OAL+
           000
1609            ****************************************************************
1610            ****************************************************************
1611            **
1612            ** Name:    BUFDAL  -  Deallocate Card Header I/Obuffer
1613            **
1614            ** Category:   LOCAL
1615            **
1616            ** Purpose:
```

```
1617                **      Deallocate card header I/Obuffer.
1618                **
1619                ** Entry:
1620                **      None.
1621                **
1622                ** Exit:
1623                **      P=0.
1624                **
1625                ** Calls:      I/ODAL (falls through).
1626                **
1627                ** Uses.......
1628                **              A,B,C,D0,D1,P.
1629                **
1630                ** Stk lvls:   2
1631                **
1632                ** History:
1633                **
1634                **    Date      Programmer              Modification
1635                **  --------    ----------   ----------------------------------
1636                **  06/25/82    NM           Added documentation
1637                **
1638                ****************************************************************
1639                ****************************************************************
1640 1CF03 20     BUFDAL P=     0
1641 1CF05 3270           LC(3)  =bCARD
           8
1642 1CF0A 8D00 =i/odal GOVLNG =I/ODAL
           000
1643                *-
1644                *-
1645 1CF11 1B12    STR1-2 D0=(5) (=SCRTCH)+32
           9F2
1646 1CF18 111            A=R1
1647 1CF1B 1507           DAT0=A W
1648 1CF1F 16F            D0=D0+ 16
1649 1CF22 112            A=R2
1650 1CF25 1507           DAT0=A W
1651 1CF29 01             RTN
1652                *
1653 1CF2B 8C00    fpoll  GOLONG =FPOLLj
           00
1654                *
1655 1CF31         =r<rst2
1656 1CF31 8D00 =savlvl GOVLNG =R<RST2        Save 3 levels.
           000
1657                *----savlvl P=     2
1658                *----       GOVLNG =R<RSTK
1659 1CF38         =rst2<r
1660 1CF38 8D00 =rstlvl GOVLNG =RST2<R
           000
1661                *----rstlvl P=     2
1662                *----       GOVLNG =RSTK<R
1663                *
```

```
1664                    STITLE Message routines
1665                ********************************************************************
1666                ********************************************************************
1667                **
1668                ** Name:    UNKCD   -  Error Messages
1669                ** Name:    RWERR   -  Error Messages
1670                ** Name:    NROOM   -  Error Messages
1671                ** Name:    FTYPER  -  Error Messages
1672                ** Name:    PLLCRD  -  PULL CARD Message
1673                **
1674                ** Category:   LOCAL
1675                **
1676                ** Purpose:
1677                **      Send message to display.
1678                **
1679                ** Entry:
1680                **      HEX mode.
1681                **
1682                ** Exit:
1683                **      PLLC: Through MESSAG.
1684                **      (all others): Through WRMSG.
1685                **
1686                ** Calls:      MESSAG or WRMSG (falls through).
1687                **
1688                ** Uses.......
1689                **              A,B,C,D,P,D0,D1,R0,scratch RAM,
1690                **
1691                ** Stk lvls:   1
1692                **
1693                ** Algorithm:
1694                **      D1[1-0]=Mainframe message number.
1695                **      Fall through to MESSAG/ERMSG.
1696                **
1697                ** History:
1698                **
1699                **    Date      Programmer              Modification
1700                **  --------   ----------   ----------------------------------
1701                **  06/22/82   NM           Added documentation
1702                **
1703                ********************************************************************
1704                ********************************************************************
1705 1CF3F 1D00   UNKCD  D1=(2) =eUNKCD
1706 1CF43 6470          GOTO   WRMSG
1707 1CF47 1D00   PLLCRD D1=(2) =ePLLC
1708 1CF4B 6C70          GOTO   MESSAG
1709 1CF4F 1D00   RWERR  D1=(2) =eRWERR
1710 1CF53 6460          GOTO   WRMSG
1711 1CF57 1D00   NROOM  D1=(2) =eMEM
1712 1CF5B 6DD0          GOTO   errout
1713 1CF5F 1D00   FTYPER D1=(2) =eFTYPE
1714 1CF63 65D0          GOTO   errout
1715                ********************************************************************
1716                ********************************************************************
1717                **
1718                ** Name:    RDYTRK  -  "Pull <trk#> Of <#trks>"
```

```
1719                ** 
1720                ** Category:   LOCAL
1721                ** 
1722                ** Purpose:
1723                **      Solicit user to pull a card.
1724                ** 
1725                ** Entry:
1726                **   (both): R2 points at card header I/Obuffer.
1727                **   RDYT10: D0[3-0] contains message #.
1728                **           P set for MFWRN routine.
1729                ** 
1730                ** Exit:
1731                **      Through MESS++.
1732                ** 
1733                ** Calls:      STR1-2, CSLW5, MESS++ (falls through).
1734                ** 
1735                ** Uses.......
1736                **           A,B,C,D,P,D0,D1,R0,scratch RAM,
1737                ** 
1738                ** Stk lvls:   1
1739                ** 
1740                ** Detail:
1741                **      Reads <trk#> and <#trks> from bytes 1 and 2 of card
1742                **      header I/Obuffer.
1743                ** 
1744                ** Algorithm:
1745                **   RDYTRK: D0[0-3]=ePLLC.
1746                **           P=7 (message type for MFWRN).
1747                **   RDYT10: Point at card header I/Obuffer.
1748                **           R2[A]=current trk# from header I/Obuffer.
1749                **           R2[9-5]=max trk# from header I/Obuffer.
1750                **           C[3-0]=message# from D0.
1751                **           C[14-13]=F4 (identifies parms in R2).
1752                **           Fall through to MESS++.
1753                ** 
1754                ** History:
1755                ** 
1756                **    Date     Programmer             Modification
1757                **  --------   ----------   ----------------------------------
1758                **  06/22/82   NM           Added documentation
1759                ** 
1760                ********************************************************************
1761                ********************************************************************
1762 1CF67 1A00  RDYTRK D0=(4) =ePLLC#
          00
1763 1CF6D 27           P=     7              Normal message
1764 1CF6F 11A   RDYT10 C=R2
1765 1CF72 135          D1=C
1766 1CF75 173          D1=D1+ 2*2            Point at max track#
1767 1CF78 136          CDOEX                 Hold d0
1768 1CF7B 729F         GOSUB  STR1-2         Stash scratch regs
1769 1CF7F 134          D0=C                  Restore d0
1770 1CF82 D2           C=0    A
1771 1CF84 14F          C=DAT1 B
1772 1CF87 80F5         CPEX   5
```

```
1773 1CF8B 8F00          GOSBVL =CSLW5
          000
1774 1CF92 1C1           D1=D1- 2              Point at track#
1775 1CF95 14F           C=DAT1 B
1776 1CF98 10A           R2=C                  Two arguments for message
1777 1CF9B 2D            P=     13
1778 1CF9D 314F          LCHEX  F4             Identify arguments
1779 1CFA1 13E           CDOXS                 Get message #
1780 1CFA4 80DA          P=C    10             Retrieve msgtype pointer
1781 1CFA8 6820          GOTO   MESS++         Display message
1782              ************************************************************
1783              ************************************************************
1784              **
1785              ** Name:    TRKDON  -  "Trk #<trk#> Done"
1786              **
1787              ** Category:   LOCAL
1788              **
1789              ** Purpose:
1790              **      Put up "Trk #xxx done" message.
1791              **
1792              ** Entry:
1793              **      R2 points at card header I/Obuffer.
1794              **
1795              ** Exit:
1796              **      Through RDYT10.
1797              **
1798              ** Calls:      RDYT10 (falls through).
1799              **
1800              ** Uses.......
1801              **             A,B,C,D,P,D0,D1,R0,scratch RAM,
1802              **
1803              ** Stk lvls:   1
1804              **
1805              ** Detail:
1806              **      Reads trk# from card header I/Obuffer.
1807              **      RDYT10 sets up as for RDYTRK message, but this message
1808              **        only uses the first number (trk#).
1809              **
1810              ** Algorithm:
1811              **      D0[0-3]=eTRKDN.
1812              **      P=6 (delay, nobeep, nostore).
1813              **      Fall through to RDYT10.
1814              **
1815              ** History:
1816              **
1817              **    Date      Programmer              Modification
1818              **  --------   ----------   ------------------------------------
1819              **  06/22/82   NM           Added documentation
1820              **
1821              ************************************************************
1822              ************************************************************
1823 1CFAC 1A00 =TRKDON D0=(4) =eTRKDN
          00
1824 1CFB2 26            P=     6
1825 1CFB4 6ABF          GOTO   RDYT10
```

```
1826            ********************************************************************
1827            ********************************************************************
1828            **
1829            ** Name:    WRMSG   -  Send Warning Message To Display
1830            **
1831            ** Category:   LOCAL
1832            **
1833            ** Purpose:
1834            **      Send a warning message to the display.
1835            **
1836            ** Entry:
1837            **      D1[B] = message#
1838            **
1839            ** Exit:
1840            **      RTNSC
1841            **
1842            ** Calls:      CRDOFF, MESS10 (falls through).
1843            **
1844            ** Uses.......
1845            **              A,B,C,D,P,D0,D1,R0,scratch RAM,
1846            **
1847            ** Stk lvls:   1
1848            **
1849            ** Algorithm:
1850            **      Turn off card reader.
1851            **      Set P=8 for MFWRN (beep, delay, ERRN, "WRN")
1852            **      MESS10.
1853            **
1854            ** History:
1855            **
1856            **    Date      Programmer              Modification
1857            **  --------   ----------   ----------------------------------
1858            **  06/22/82   NM           Added documentation
1859            **
1860            ********************************************************************
1861            ********************************************************************
1862 1CFB8 133   WRMSG  AD1EX
1863 1CFBB 74F0         GOSUB  CRDOFF
1864 1CFBF 131          D1=A                    Restore message pointer
1865 1CFC2 28           P=     8                Beep, delay
1866 1CFC4 6500         GOTO   MESS10
1867            ********************************************************************
1868            ********************************************************************
1869            **
1870            ** Name:    MESSAG  -  Send Message To Display
1871            **
1872            ** Category:   LOCAL
1873            **
1874            ** Purpose:
1875            **      Send message to display; erroring out if battery low.
1876            **
1877            ** Entry:
1878            **   MESSAG: D1[B] = message#
1879            **   MESS10: D1[B] = message#
1880            **           P set for MFWRN.
```

```
1881                ** MESS++: Scratch registers already stashed (STR1-2).
1882                **         A[3-0] = message#.
1883                **         P set for MFWRN.
1884                **
1885                ** Exit:
1886                **      Carry set.
1887                **
1888                ** Calls:      ACBAT?, MFWRN, STR1-2, Sflag?, atnclr, rstlvl,
1889                **             savlvl.
1890                **
1891                ** Uses.......
1892                **             A,B,C,D,P,D0,D1,R0,scratch RAM,
1893                **
1894                ** Stk lvls:   1
1895                **
1896                ** Algorithm:
1897                **   MESSAG: P=7 (nowarn, nodelay, nobeep).
1898                **   MESS10: Stash scratch regs.
1899                **           A[0-3]=00[D0[B]].
1900                **   MESS++: Stash 3 subroutine levels in R3.
1901                **           Stash errmsg# and ptr in R1.
1902                **           Check battery level.  "eLOBAT" if low battery.
1903                **           Fetch errmsg# and ptr from R1.
1904                **           MFWRN.
1905                **           Restore 3 subroutine levels from R3.
1906                **           Restore scratch regs.
1907                **           Return with CARRY SET.
1908                **
1909                ** History:
1910                **
1911                **    Date      Programmer            Modification
1912                **  --------   ----------   ----------------------------------
1913                **  06/22/82   NM           Added documentation
1914                **  09/08/82   NM           Added battery check code
1915                **
1916                ********************************************************************
1917                ********************************************************************
1918 1CFC8 27    MESSAG P=     7              Nowarn, no delay
1919 1CFCA 734F  MESS10 GOSUB  STR1-2         Stash scratch regs
1920 1CFCE 13F          CD1XS                 Get message number
1921 1CFD1 80FF  MESS++ CPEX   15             Stash ptr in C[4]
1922 1CFD5 109          R1=C                  Save message info
1923 1CFD8 8E00         GOSUBL =atnclr        Clear ATTN flag
           00
1924 1CFDE 7F4F         GOSUB  savlvl         Stash 3 levels
1925 1CFE2 8F00         GOSBVL =ACBAT?        Check battery level
           000
1926 1CFE9 313C         LC(2)  =flBAT
1927 1CFED 8E00         GOSUBL =Sflag?        Low battery?
           00
1928 1CFF3 531          GONC   MESS20         No.
1929 1CFF6 20           P=     0              Yes.
1930 1CFF8 3300         LC(4)  =eLOBAT
           00
1931 1CFFE 28           P=     8
```

```
1932 1D000 8F00          GOSBVL =MFWRN          Warn user.
           000
1933 1D007 119    MESS20 C=R1                   Fetch message info.
1934 1D00A 22            P=     2
1935 1D00C 3100          LCHEX  00              Clear upper byte of msg#
1936 1D010 80DF          P=C    15              Restore ptr for message
1937 1D014 8F00          GOSBVL =MFWRN          Display msg, using lots of levels
           000
1938 1D01B 791F          GOSUB  rstlvl          Restore 3 levels
1939 1D01F 1B12          DO=(5) (=SCRTCH)+32
           9F2
1940 1D026 1527          A=DATO W
1941 1D02A 101           R1=A
1942 1D02D 16F           DO=DO+ 16
1943 1D030 1527          A=DATO W
1944 1D034 102           R2=A
1945 1D037 02            RTNSC
1946               *
1947 1D039 137    errout CD1EX
1948 1D03C 10B           R3=C                   Stash message number
1949 1D03F 70CE          GOSUB  BUFDAL          Deallocate I/O buffer
1950 1D043 11B           C=R3                   Recall message #
1951 1D046 8C00          GOLONG =MFERRj         Error out
           00
1952               *
1953               ***************************************************************
1954               ***************************************************************
1955               **
1956               ** Name:    RALIGN  -  Send "align" Message To Display
1957               **
1958               ** Category:   LOCAL
1959               **
1960               ** Purpose:
1961               **      Send "align card, then ENDLN" message to display
1962               **      prefixed by operation descriptor (WRT, READ, etc.).
1963               **
1964               ** Entry:
1965               **      None.
1966               **
1967               ** Exit:
1968               **      If carry clear, user hit ENDLN.
1969               **      If carry set, we abort (timeout or user hit ATTN).
1970               **
1971               ** Calls:      FINDAj, MESSAG, SCRLLR, nokeys, rstlvl, savlvl.
1972               **
1973               ** Uses.......
1974               **             A,B,C,D,P,D0,D1,R0,scratch RAM
1975               **
1976               ** Stk lvls:   3
1977               **
1978               ** History:
1979               **
1980               **    Date     Programmer             Modification
1981               **  --------   ----------   ------------------------------------
1982               **  06/22/82   NM           Added documentation
```

```
1983                **
1984                ********************************************************************
1985                ********************************************************************
1986 1D04C 1D00    RALIGN D1=(2) =eRALGN
1987 1D050 747F    ALIGN  GOSUB  MESSAG
1988 1D054 7920    ALIG10 GOSUB  nokeys          Clear keybuffer
1989 1D058 75DE           GOSUB  savlvl
1990 1D05C 8F00           GOSBVL =SCRLLR         Sleep with display scroll
           000
1991 1D063 71DE           GOSUB  rstlvl
1992 1D067 8E00           GOSUBL =FINDAj
           00
1993 1D06D 62             CON(2) =k#EOL          If Endline
1994 1D06F 210            REL(3) ALIG25          Clear buffer, RTNCC
1995 1D072 B2             CON(2) =k#ATTN         If ATTN
1996 1D074 410            REL(3) ALIG35          Clear buffer, RTNSC
1997 1D077 36             CON(2) =k#OFF          If OFF
1998 1D079 F00            REL(3) ALIG35          Clear buffer, RTNSC
1999 1D07C 00             CON(2) 0               Else
2000 1D07E 55D            GONC   ALIG10          Try again.  B.E.T.
2001 1D081         ALIG25
2002 1D081 8D00    nokeys GOVLNG =NOKEYS
           000
2003 1D088         ALIG35
2004 1D088 75FF           GOSUB  nokeys
2005 1D08C 02             RTNSC
```

```
2006                    STITLE Hardware interface
2007                ****************************************************************
2008                **
2009                **   Card reader is hard-configured at 2C000, known as
2010                **   (=CR).  following locations contain following stuff:
2011                **
2012                **     (=CR) to (=CR)+#F: FIFO buffer I/O.
2013                **     (=CR)+#10: Request enable bits for (=CR)+#12.
2014                **     (=CR)+#11: Request enable bits for (=CR)+#13.
2015                **     (=CR)+#12: Bit 3--FIFO full/empty (on read/write),
2016                **                Bit 2--too fast,
2017                **                Bit 1--too slow,
2018                **                Bit 0--data error.
2019                **                Bits enabled to request line by (=CR)+#10.
2020                **     (=CR)+#13: Bit 3--4 bytes ready in FIFO,
2021                **                Bit 2--3 bytes ready in FIFO,
2022                **                Bit 1--2 bytes ready in FIFO,
2023                **                Bit 0--1 byte ready in FIFO.
2024                **                Bits enabled to request line by (=CR)+#11.
2025                **     (=CR)+#14: Bit 3--chip awake,
2026                **                Bit 2--write mode,
2027                **                Bit 1--flagserve (breakserve),
2028                **                Bit 0--"1"   (always set).
2029                **     (=CR)+#15: test bits for chip testing.  leave #F!!
2030                **
2031                ****************************************************************
2032                *
2033                ****************************************************************
2034                ****************************************************************
2035                **
2036                ** Name:    WRITE   -  Set Card Reader Write Mode
2037                **
2038                ** Category:   LOCAL
2039                **
2040                ** Purpose:
2041                **      Set card reader write mode.
2042                **
2043                ** Entry:
2044                **   WRITE: None.
2045                **   WRITCP: C[P] = nibble to write to CR control register.
2046                **
2047                ** Exit:
2048                **      Carry clear.
2049                **
2050                ** Calls:      None.
2051                **
2052                ** Uses.......
2053                **             C[P] (whatever P is),C[3-0],D1.
2054                **
2055                ** Stk lvls:   0
2056                **
2057                ** Algorithm:
2058                **   WRITE: C[P] = C.
2059                **   WRITCP: Write C[P] to (=CR)+#14.
2060                **           Read FIFO and request bits to flush.
```

```
2061                **
2062                ** History:
2063                **
2064                **    Date     Programmer          Modification
2065                **  --------   ----------  ---------------------------------
2066                **  06/22/82   NM          Added documentation
2067                **
2068                ********************************************************************
2069                ********************************************************************
2070 1D08E 30C     WRITE  LCHEX  C              Chip awake and write mode
2071 1D091 1F41    WRITCP D1=(5) (=CR)+#14      Control register write
           0C2
2072 1D098 1550          DAT1=C P
2073 1D09C 1D00          D1=(2) (=CR)
2074 1D0A0 1577          C=DAT1 W              Read fifo to flush
2075 1D0A4 17F           D1=D1+ 16
2076 1D0A7 AE2           C=0    B
2077 1D0AA 14D           DAT1=C B              Clear rqst enable bits
2078 1D0AD 15F3          C=DAT1 4              Read rqst bits to flush
2079 1D0B1 03            RTNCC
2080                ********************************************************************
2081                ********************************************************************
2082                **
2083                ** Name:    CRDOFF  -  Turn Off Card Reader
2084                **
2085                ** Category:   LOCAL
2086                **
2087                ** Purpose:
2088                **      Turn off card reader.
2089                **
2090                ** Entry:
2091                **      None.
2092                **
2093                ** Exit:
2094                **      RTNCC (through WRITCP).
2095                **
2096                ** Calls:      None.
2097                **
2098                ** Uses.......
2099                **             C[P],C[3-0],D1.
2100                **
2101                ** Stk lvls:   0
2102                **
2103                ** Algorithm:
2104                **      C[P] = 0.
2105                **      WRITCP.
2106                **
2107                ** History:
2108                **
2109                **    Date     Programmer          Modification
2110                **  --------   ----------  ---------------------------------
2111                **  06/22/82   NM          Added documentation
2112                **
2113                ********************************************************************
2114                ********************************************************************
```

```
1DOB3 A82   CRDOFF C=0    P
1DOB6 6ADF         GOTO   WRITCP
              ****************************************************************
              ****************************************************************
              **
              ** Name:    READFL  -  Set BRKSRV In Read Mode
              **
              ** Category:   LOCAL
              **
              ** Purpose:
              **      Set BRKSRV (FLGSRV) in read mode.
              **
              ** Entry:
              **      None.
              **
              ** Exit:
              **      Carry clear (through WRITCP).
              **
              ** Calls:      None.
              **
              ** Uses.......
              **             C[P],C[3-0],D1.
              **
              ** Stk lvls:   0
              **
              ** Algorithm:
              **      C[P]=A.
              **      WRITCP.
              **
              ** History:
              **
              **    Date     Programmer              Modification
              **  --------   ----------   ----------------------------------
              **  06/22/82   NM           Added documentation
              **
              ****************************************************************
              ****************************************************************
1DOBA 30A   READFL LCHEX  A
1DOBD 63DF         GOTO   WRITCP
              ****************************************************************
              ****************************************************************
              **
              ** Name:    WRITFL  -  Set BRKSRV In Write Mode
              **
              ** Category:   LOCAL
              **
              ** Purpose:
              **      Set BRKSRV in write mode.
              **
              ** Entry:
              **      None.
              **
              ** Exit:
              **      Carry clear (through WRITCP).
              **
```

```
2170            ** Calls:      WAITMT
2171            **
2172            ** Uses.......
2173            **             C[5-0],P,D1,B[B],S1
2174            **             {A-D,D0,D1,P,R0,scratch RAM if error}
2175            **
2176            ** Stk lvls:   1
2177            **
2178            ** Algorithm:
2179            **      Wait until buffer empty (WAITMT).
2180            **      C[P]=E.
2181            **      WRITCP.
2182            **
2183            ** History:
2184            **
2185            **    Date      Programmer              Modification
2186            **  --------   ----------   ----------------------------------
2187            **  06/22/82   NM           Added documentation
2188            **
2189            ************************************************************************
2190            ************************************************************************
2191 1D0C1 841   WRITFL ST=0   1
2192 1D0C4 7670         GOSUB  WAITMT
2193 1D0C8 30E          LCHEX  E
2194 1D0CB 65CF         GOTO   WRITCP
2195            ************************************************************************
2196            ************************************************************************
2197            **
2198            ** Name:    READ8   -  Read 8 Nibbles From Card W/o Slow Chk
2199            **
2200            ** Category:   LOCAL
2201            **
2202            ** Purpose:
2203            **      Read 8 nibbles at beginning of card (i.e., we are
2204            **      looking before card starts moving).
2205            **
2206            ** Entry:
2207            **      None.
2208            **
2209            ** Exit:
2210            **      Carry set if failure (WAIT routine reported error
2211            **        already).
2212            **      Else result of read is in A[7-0].
2213            **
2214            ** Calls:      WAITRQ.
2215            **
2216            ** Uses.......
2217            **             A[7-0],C[5-0],P,D1,B[B],S1
2218            **             {A-D,D0,D1,P,R0,scratch RAM if error}
2219            **
2220            ** Stk lvls:   1
2221            **
2222            ** Detail:
2223            **      This is used only for the first read on the card.
2224            **      After said read, the code must look out for too slow
```

```
2225            **      error--using the READ8S routine.
2226            **
2227            ** Algorithm:
2228            **      WAITRQ {if error, WAIT throws off a subroutine level
2229            **              and returns to routine which called this one}.
2230            **      Read 8 nibbles from FIFO into A.
2231            **      RTNCC.
2232            **
2233            ** History:
2234            **
2235            **    Date      Programmer              Modification
2236            **  --------    ----------   ----------------------------------
2237            **  06/22/82    NM           Added documentation
2238            **
2239            ********************************************************************
2240            ********************************************************************
2241 1D0CF 841   READ8  ST=0   1
2242 1D0D2 7A30         GOSUB  WAITRQ         Wait for bufrequest
2243 1D0D6 1587         A=DAT1 8
2244 1D0DA 03           RTNCC
2245            ********************************************************************
2246            ********************************************************************
2247            **
2248            ** Name:    READ8S  -  Read 8 Nibbles From Card
2249            **
2250            ** Category:   LOCAL
2251            **
2252            ** Purpose:
2253            **      Read 8 nibbles from card.
2254            **
2255            ** Entry:
2256            **      None.
2257            **
2258            ** Exit:
2259            **      Carry set if failure (WAIT already reported error).
2260            **      Else 8 nibbles of data in A[7-0].
2261            **
2262            ** Calls:      WAITRS.
2263            **
2264            ** Uses.......
2265            **              A[7-0],C[5-0],P,D1,B[B],S1
2266            **              {A-D,D0,D1,P,R0,scratch RAM if error}
2267            **
2268            ** Stk lvls:   1
2269            **
2270            ** Detail:
2271            **      If WAIT finds an error, it will report the error and
2272            **      RTNSC to the routine which called this one.
2273            **
2274            ** Algorithm:
2275            **      WAITRS.
2276            **      Read 8 nibbles at FIFO.
2277            **      RTNCC.
2278            **
2279            ** History:
```

```
2280                **
2281                **    Date     Programmer          Modification
2282                **  --------   ----------  ----------------------------------
2283                **  06/22/82   NM          Added documentation
2284                **
2285                ************************************************************
2286                ************************************************************
2287 1D0DC 841    READ8S ST=0   1
2288 1D0DF 7730   RD8SV  GOSUB  WAITRS
2289 1D0E3 15B7         A=DAT1 8
2290 1D0E7 03           RTNCC
2291                ************************************************************
2292                ************************************************************
2293                **
2294                ** Name:    WRIT8S  -  Write 8 Nibbles To Card
2295                **
2296                ** Category:   LOCAL
2297                **
2298                ** Purpose:
2299                **      Write 8 nibbles of data to card.
2300                **
2301                ** Entry:
2302                **      Data in A[7-0].
2303                **
2304                ** Exit:
2305                **      RTNSC if failure (WAIT reported error).
2306                **      RTNCC if wrote successfully.
2307                **
2308                ** Calls:      WAITRS.
2309                **
2310                ** Uses.......
2311                **             C[5-0],P,D1,B[B],S1
2312                **             {A-D,D0,D1,P,R0,scratch RAM if error}
2313                **
2314                ** Stk lvls:   1
2315                **
2316                ** Algorithm:
2317                **      WAITRS.
2318                **      Write 8 nibbles to FIFO.
2319                **      RTNCC.
2320                **
2321                ** History:
2322                **
2323                **    Date     Programmer          Modification
2324                **  --------   ----------  ----------------------------------
2325                **  06/22/82   NM          Added documentation
2326                **
2327                ************************************************************
2328                ************************************************************
2329 1D0E9 841    WRIT8S ST=0   1
2330 1D0EC 7A20         GOSUB  WAITRS
2331 1D0F0 1597         DAT1=A 8
2332 1D0F4 03           RTNCC
2333                ************************************************************
2334                ************************************************************
```

```
2335                **
2336                ** Name:    READCS  -  Read FIFO When < 8 Nibbles Ready
2337                **
2338                ** Category:   LOCAL
2339                **
2340                ** Purpose:
2341                **      Read end-of-field on card when < 8 nibs long.
2342                **
2343                ** Entry:
2344                **      C[S] = #nibbles to expect {allow 1,3,5 only}.
2345                **
2346                ** Exit:
2347                **      Carry set if failure.
2348                **      Else result in A[0-{original C[S]}].
2349                **
2350                ** Calls:      WAITCS.
2351                **
2352                ** Uses.......
2353                **              A[7-0],C[5-0],P,D1,B[B],S1
2354                **              {A-D,D0,D1,P,R0,scratch RAM if error}
2355                **
2356                ** Stk lvls:   1
2357                **
2358                ** Detail:
2359                **      Used to read FIFO when only 1, 2 or 3 bytes are ready.
2360                **      Needed if last read in a field is < 8 nibbles long...
2361                **      read is performed when data is in, before hitting post-
2362                **      field crap on card.  Performs a 4-byte (8-nibble) read
2363                **      regardless of size which was requested.
2364                **
2365                ** Algorithm:
2366                **      WAITCS.
2367                **      Read 8 nibbles from FIFO.
2368                **      RTNCC.
2369                **
2370                ** History:
2371                **
2372                **    Date      Programmer              Modification
2373                **   --------   ----------   ----------------------------------
2374                **   06/22/82   NM           Added documentation
2375                **
2376                ********************************************************************
2377                ********************************************************************
2378 1D0F6 841     READCS ST=0   1
2379 1D0F9 7720           GOSUB  WAITCS
2380 1D0FD 15B7           A=DAT1 8
2381 1D101 03             RTNCC
2382                ********************************************************************
2383                ********************************************************************
2384                **
2385                ** Name:    WRIT4S  -  Write 4 Nibbles To Card
2386                **
2387                ** Category:   LOCAL
2388                **
2389                ** Purpose:
```

```
2390                **      Write 2 bytes of data to card.
2391                **
2392                ** Entry:
2393                **      Data to write in A[3-0].
2394                **
2395                ** Exit:
2396                **      RTNCC if successful.
2397                **      Else WAIT reported error.
2398                **
2399                ** Calls:      WAITRS.
2400                **
2401                ** Uses.......
2402                **              C[5-0],P,D1,B[B],S1
2403                **              {A-D,D0,D1,P,R0,scratch RAM if error}
2404                **
2405                ** Stk lvls:   1
2406                **
2407                ** Detail:
2408                **      Used by protect/unprotect to finish off the write of
2409                **      that ridiculously-shaped field.
2410                **
2411                ** History:
2412                **
2413                **    Date      Programmer              Modification
2414                **  --------    ----------   ----------------------------------
2415                **  06/22/82    NM           Added documentation
2416                **
2417                ********************************************************************
2418                ********************************************************************
2419 1D103 841    WRIT4S ST=0   1
2420 1D106 7010         GOSUB  WAITRS
2421 1D10A 1593         DAT1=A 4
2422 1D10E 03           RTNCC
2423                ********************************************************************
2424                ********************************************************************
2425                **
2426                ** Name:    WAITRQ  -  Wait For BUFREQ Or TUFAST
2427                **
2428                ** Category:   LOCAL
2429                **
2430                ** Purpose:
2431                **      Wait for BUFREQ (non-error condition) or TUFAST (error
2432                **      condition.
2433                **
2434                ** Entry:
2435                **      None.
2436                **
2437                ** Exit:
2438                **      If error condition occurs, sends error message to
2439                **        display, throws off a subroutine level and returns
2440                **        with carry set.
2441                **      Else carry clear, D1 points at FIFO.
2442                **
2443                ** Calls:      WAIT (falls through).
2444                **
```

```
2445            ** Uses.......
2446            **              C[5-0],P,D1,B[B],S1
2447            **              {A-D,D0,D1,P,R0,scratch RAM if error}
2448            **
2449            ** Stk lvls:    0
2450            **
2451            ** Detail:
2452            **      Because of error return, this cannot be called from
2453            **      the top level of code.  Typically it is called from
2454            **      a read or write routine which was called from the top
2455            **      level of code.
2456            **
2457            ** Algorithm:
2458            **      Enable BUFREQ and TUFAST to request line.
2459            **      WAIT.
2460            **
2461            ** History:
2462            **
2463            **    Date      Programmer              Modification
2464            **  --------   ----------   ----------------------------------
2465            **  06/23/82   NM           Added documentation
2466            **
2467            ********************************************************************
2468            ********************************************************************
2469 1D110 20     WAITRQ P=     0
2470            TEST   IF      TEST
2472            TEST   ELSE
2473 1D112 3148        LCHEX  84              Enable bufreq & tufast
2474            TEST   ENDIF
2475 1D116 6D20        GOTO   WAIT
2476            ********************************************************************
2477            ********************************************************************
2478            **
2479            ** Name:    WAITRS  -  Wait For BUFREQ Or Error Conditions
2480            **
2481            ** Category:   LOCAL
2482            **
2483            ** Purpose:
2484            **      Wait for BUFREQ (non-error condition) or TUFAST,
2485            **      TUSLOW or DATERR (error condition).
2486            **
2487            ** Entry:
2488            **      None.
2489            **
2490            ** Exit:
2491            **      If error condition occurs, sends error message to
2492            **        display, throws off a subroutine level and returns
2493            **        with carry set.
2494            **      Else carry clear, D1 points at FIFO.
2495            **
2496            ** Calls:      WAIT (falls through).
2497            **
2498            ** Uses.......
2499            **              C[5-0],P,D1,B[B],S1
2500            **              {A-D,D0,D1,P,R0,scratch RAM if error}
```

```
2501            **
2502            ** Stk lvls:   0
2503            **
2504            ** Detail:
2505            **      Because of error return, this cannot be called from
2506            **      the top level of code.  Typically it is called from
2507            **      a read or write routine which was called from the top
2508            **      level of code.
2509            **
2510            ** Algorithm:
2511            **      Enable BUFREQ, TUFAST, TUSLOW, DATERR to request line.
2512            **      WAIT.
2513            **
2514            ** History:
2515            **
2516            **    Date     Programmer             Modification
2517            **  --------   ----------   ----------------------------------
2518            **  06/23/82   NM           Added documentation
2519            **
2520            ************************************************************************
2521            ************************************************************************
2522 1D11A 20     WAITRS P=     0
2523            TEST   IF     TEST
2525            TEST   ELSE
2526 1D11C 3178         LCHEX  87             Enable bufreq & error bits
2527            TEST   ENDIF
2528 1D120 6320         GOTO   WAIT
2529            ************************************************************************
2530            ************************************************************************
2531            **
2532            ** Name:    WAITCS  -  Wait For Ready For Partial Read
2533            **
2534            ** Category:   LOCAL
2535            **
2536            ** Purpose:
2537            **      Wait for 1, 2 or 3 bytes ready (non-error condition)
2538            **      or TUFAST, TUSLOW, or DATERR (error conditions).
2539            **
2540            ** Entry:
2541            **      C[S] = #nibbles-1 to wait for.  Only allowed values
2542            **        are 1,3,5.  Anything else produces unpredictable
2543            **        results.
2544            **
2545            ** Exit:
2546            **      If error condition occurs, sends error message to
2547            **        display, throws off a subroutine level and returns
2548            **        with carry set.
2549            **      Else carry clear, D1 points at FIFO.
2550            **
2551            ** Calls:      WAIT (falls through).
2552            **
2553            ** Uses.......
2554            **              C,P,D1,B[B],
2555            **              {A-D,D0,D1,P,R0,scratch RAM if error}
2556            **
```

```
2557            ** Stk lvls:   0
2558            **
2559            ** Detail:
2560            **      Because of error return, this cannot be called from
2561            **      the top level of code.  Typically it is called from
2562            **      a read routine which was called from the top level of
2563            **      code.
2564            **
2565            ** Algorithm:
2566            **      Case C[S] of
2567            **        1: Enable 1-byte-ready to request line.
2568            **        3: Enable 2-bytes-ready to request line.
2569            **        5: Enable 3-bytes-ready to request line.
2570            **      End case.
2571            **      Enable TUFAST, TUSLOW, DATERR to request line.
2572            **      WAIT.
2573            **
2574            ** History:
2575            **
2576            **    Date     Programmer              Modification
2577            **  --------   ----------   ----------------------------------
2578            **  06/23/82   NM           Added documentation
2579            **
2580            ****************************************************************
2581            ****************************************************************
2582 1D124 BCA   WAITCS C=-C   S
2583 1D127 80DF        P=C    15
2584             TEST   IF     TEST
2586             TEST   ELSE
2587 1D12B 3607        LCHEX  4727170        Set rqst & error bits
           1727
           4
2588             TEST   ENDIF
2589 1D134 6F00        GOTO   WAIT
2590            ****************************************************************
2591            ****************************************************************
2592            **
2593            ** Name:    WAITM+  -  Call WAITMT
2594            **
2595            ** Category:   LOCAL
2596            **
2597            ** Purpose:
2598            **      Provide call to WAITMT.
2599            **
2600            ** Entry:
2601            **      None.
2602            **
2603            ** Exit:
2604            **      If carry set, error condition occurred and WAIT put up
2605            **        error message.
2606            **      Else carry clear, D1 points at FIFO.
2607            **
2608            ** Calls:      WAIT (falls through).
2609            **
2610            ** Uses.......
```

```
2611            **              C[5-0],P,D1,B[B],
2612            **              {A-D,D0,D1,P,R0,scratch RAM if error}
2613            **
2614            ** Stk lvls:    1
2615            **
2616            ** Detail:
2617            **      This routine is used since WAITMT cannot be called from
2618            **      the top level of code.
2619            **
2620            ** Algorithm:
2621            **      WAITMT.
2622            **      RTN.
2623            **
2624            ** History:
2625            **
2626            **    Date      Programmer              Modification
2627            **  --------    ----------   ----------------------------------
2628            **  06/23/82    NM           Added documentation
2629            **
2630            ********************************************************************
2631            ********************************************************************
2632 1D138 7200  WAITM+ GOSUB  WAITMT         In case of error, since subr.
2633 1D13C 01           RTN                   Level is ejected
2634            ********************************************************************
2635            ********************************************************************
2636            **
2637            ** Name:     WAITMT  -  Wait For Buffer Empty/full Or Error
2638            **
2639            ** Category:   LOCAL
2640            **
2641            ** Purpose:
2642            **      Wait for buffer full/empty (non-error condition) or
2643            **      TUSLOW, TUFAST, DATERR (error conditions).
2644            **
2645            ** Entry:
2646            **      None.
2647            **
2648            ** Exit:
2649            **      If error condition occurs, sends error message to
2650            **        display, throws off a subroutine level and returns
2651            **        with carry set.
2652            **      Else carry clear, D1 points at FIFO.
2653            **
2654            ** Calls:      WAIT (falls through).
2655            **
2656            ** Uses.......
2657            **              C[5-0],P,D1,B[B],
2658            **              {A-D,D0,D1,P,R0,scratch RAM if error}
2659            **
2660            ** Stk lvls:    0
2661            **
2662            ** Detail:
2663            **      Because of error return, this cannot be called from
2664            **      the top level of code.  Typically it is called from
2665            **      a read or write routine which was called from the top
```

```
2666            **      level of code.
2667            **
2668            ** Algorithm:
2669            **      Enable all of CRRQS1 to request line.
2670            **      WAIT.
2671            **
2672            ** History:
2673            **
2674            **    Date     Programmer          Modification
2675            **   --------  ----------  ----------------------------------
2676            **   06/23/82  NM          Added documentation
2677            **
2678            ********************************************************************
2679            ********************************************************************
2680 1D13E 20    WAITMT P=     0
2681             TEST   IF     TEST
2683             TEST   ELSE
2684 1D140 31F0         LCHEX  0F             Enable full/empty & error bits
2685             TEST   ENDIF
2686            ********************************************************************
2687            ********************************************************************
2688            **
2689            ** Name:    WAIT    -  Wait For SREQ From Card Reader
2690            **
2691            ** Category:   LOCAL
2692            **
2693            ** Purpose:
2694            **      Wait for card reader ready or error condition.
2695            **
2696            ** Entry:
2697            **      C[B] = bits to enable in CRRQS1 and CRRQS2.
2698            **      S1 set iff in verify.
2699            **
2700            ** Exit:
2701            **      If error condition is found, the error is reported, a
2702            **        subroutine level ejected, and the carry set on
2703            **        return.
2704            **      EXCEPTION: If in verify (S1 set) and data error occurs
2705            **        or too-slow with BRKSRV set then RTNCC with S1=0.
2706            **      Else carry clear; D1 points at FIFO (=CR); P=0.
2707            **
2708            ** Calls:      WRMSG (only on error condition--falls through).
2709            **
2710            ** Uses.......
2711            **              C[5-0],P,D1,B[B],S1.
2712            **              {A-D,D0,D1,P,R0,scratch RAM if error}
2713            **
2714            ** Stk lvls:   0
2715            **
2716            ** Detail:
2717            **      The following bits of C[B] enable the following
2718            **      conditions:
2719            **
2720            **       Bit  Name          Condition
2721            **       ---  ----          ---------
```

```
2722            **        0   DATERR        Data error (error).
2723            **        1   TUSLOW        Too slow (error).
2724            **        2   TUFAST        Too fast (error).
2725            **        3   Over/Under    FIFO over-/under-flow.
2726            **        4   BYTE1         1 byte ready.
2727            **        5   BYTE2         2 bytes ready.
2728            **        6   BYTE3         3 bytes ready.
2729            **        7   BUFREQ        4 bytes ready.
2730            **
2731            **      Because of the error return, WAIT cannot be called from
2732            **      the top level of code.  Typically it is called from a
2733            **      read/write routine which is called from the top level
2734            **      of code.
2735            **
2736            **      Certain errors are interpreted as "verify failure" if
2737            **      we are verifying.  S1 indicates on entry that we are
2738            **      in verify.  If it is set, the following conditions
2739            **      will not cause an error message to be generated, but
2740            **      they will cause S1 to be cleared, indicating to caller
2741            **      that a verify failure occurred:
2742            **        DATA ERROR.
2743            **        TOO SLOW while BRKSRV set.
2744            **
2745            ** Algorithm:
2746            **      Write enable bits to RQEN1-RQEN2.
2747            **      Set master display driver timer to 7.5 seconds.
2748            **    1: Read enable bits (RQEN1-RQEN2) and request bits
2749            **         (CRRQS1-CRRQS2).
2750            **      AND enable bits with request bits.
2751            **      If any error bits set then goto 2.
2752            **      If timed out then report eRWERR; eject RSTK level and
2753            **        RTNSC.
2754            **    2: If bit 2 set, send "too fast" message; eject RSTK level
2755            **         and RTNSC.
2756            **      If in VERIFY goto 4.
2757            **    3: If bit 1 set, send "too slow" message; eject RSTK level
2758            **         and RTNSC.
2759            **      If bit 0 set, send "data error" message; eject RSTK
2760            **        level and RTNSC.
2761            **      If any of bits 3-7 set, RTNCC.
2762            **      Goto 1.
2763            **    4: If bit 1 set and BRKSRV clear then goto 3.
2764            **      Clear verify flag; RTNCC.
2765            **
2766            ** History:
2767            **
2768            **    Date     Programmer            Modification
2769            **  --------   ----------   ----------------------------------
2770            **  06/23/82   NM           Added documentation
2771            **
2772            ********************************************************************
2773            ********************************************************************
2774            *
2775            * NOTE: ABOVE CODE FALLS INTO THIS
2776            *
```

```
2777 1D144 1F01  WAIT   D1=(5) (=CR)+16
           0C2
2778 1D14B 14D          DAT1=C B              Write enable bits
2779 1D14E 1F8F         D1=(5) =TIMER1
           3E2
2780 1D155 26           P=     6
2781 1D157 A92   WAITDL C=0    WP
2782 1D15A A2E          C=C-1  XS
2783 1D15D 15D5         DAT1=C 6              7.5 sec timeout
2784 1D161 15F5         C=DAT1 6              Read it back
2785 1D165 B26          C=C+1  XS             C[5-0]=000000 if wrote right
2786 1D168 91E          ?C#0   WP             Wrote OK?
2787 1D16B CE           GOYES  WAITDL         No.  Try again.
2788 1D16D 20           P=     0
2789 1D16F 1F01  WTLOOP D1=(5) (=CR)+16
           0C2
2790 1D176 147          C=DAT1 A
2791 1D179 AE5          B=C    B
2792 1D17C F6           CSR    A
2793 1D17E F6           CSR    A
2794 1D180 0E61         B=C&B  B
2795 1D184 307          LCHEX  7
2796 1D187 0E05         C=B&C  P
2797 1D18B 90E          ?C#0   P              Any error bits?
2798 1D18E 72           GOYES  CERR           Yes
2799 1D190 96D          ?B#0   B              No.  other req bits set?
2800 1D193 D1           GOYES  WAITCC
2801 1D195 1FDF         D1=(5) (=TIMER1)+5
           3E2
2802 1D19C 1570         C=DAT1 P              Assume kchunk won't mess this up.
2803 1D1A0 A06          C=C+C  P              Timed out?
2804 1D1A3 5BC          GONC   WTLOOP         No
2805 1D1A6 1D00  DTERR  D1=(2) =eRWERR        "R/W error"
2806            * Following may be messy and unstructured, but it saves
2807            * Valuable subroutine stack and code.
2808 1D1AA 07    WAITER C=RSTK                Throw off calling routine
2809 1D1AC 6B0E        GOTO   WRMSG          Display error message
2810 1D1B0 1CF   WAITCC D1=D1- 16
2811 1D1B3 03           RTNCC
2812 1D1B5 1D00  CERR   D1=(2) =eTUFAS
2813 1D1B9 A06          C=C+C  P
2814 1D1BC A06          C=C+C  P
2815 1D1BF 4AE          GOC    WAITER         Go if "too fast"
2816 1D1C2 871          ?ST=1  1              Are we in verify?
2817 1D1C5 F0           GOYES  VFYER?         Yes.
2818 1D1C7 A06          C=C+C  P              No.  Too slow?
2819 1D1CA 1D00  TUSLO? D1=(2) =eTUSLO
2820 1D1CE 4BD          GOC    WAITER         Too slow.
2821 1D1D1 54D          GONC   DTERR          Data error.
2822 1D1D4 1F41  VFYER? D1=(5) (=CR)+#14       Point at CRCNTL
           0C2
2823 1D1DB 1572         C=DAT1 XS
2824 1D1DF A26          C=C+C  XS
2825 1D1E2 A26          C=C+C  XS
2826 1D1E5 A26          C=C+C  XS             Carry set if BRKSRV.
```

```
2827 1D1E8 480          GOC    VFYER!          This IS a verify error.
2828 1D1EB A06          C=C+C  P               Too slow?
2829 1D1EE 4BD          GOC    TUSLO?          Yes.  Report as too slow.
2830 1D1F1 841   VFYER!  ST=0   1               Report as verify error.
2831 1D1F4 03           RTNCC
2832              ************************************************************
2833              ************************************************************
2834              **
2835              ** Name:    WRT2-0  -  Write 16 Nibbles Of Zeroes
2836              **
2837              ** Category:   LOCAL
2838              **
2839              ** Purpose:
2840              **      Write 16 nibbles of zeroes.
2841              **
2842              ** Entry:
2843              **
2844              ** Exit:
2845              **      Carry set iff failure on write.
2846              **
2847              ** Calls:      WRIT8S.
2848              **
2849              ** Uses.......
2850              **              A,C[5-0],P,D1,B[B],S1
2851              **              {A,B,C,D,D0,R0,scratch RAM}
2852              **
2853              ** Stk lvls:   2
2854              **
2855              ** Detail:
2856              **      Used to write zeroes between fields.
2857              **
2858              ** History:
2859              **
2860              **    Date      Programmer              Modification
2861              **  --------   ----------   ---------------------------------
2862              **  06/23/82   NM           Added documentation
2863              **
2864              ************************************************************
2865              ************************************************************
2866 1D1F6 AF0   WRT2-0 A=0    W
2867 1D1F9 7CEE         GOSUB  WRIT8S
2868 1D1FD 400          RTNC
2869 1D200 68EE         GOTO   WRIT8S
2870              *
2871 1D204 8D00 =csrw5  GOVLNG =CSRW5
           000
2872 1D20B 8D00  aslw5  GOVLNG =ASLW5
           000
2873 1D212 8D00  asrw5  GOVLNG =ASRW5
           000
2874 1D219 62CE  RD8S   GOTO   READ8S
2875              *
```

```
2876                    STITLE Copy card
2877            ********************************************************************
2878            ********************************************************************
2879            **
2880            ** Name:(S) CRDFIL  -  Copy Card Into RAM
2881            **
2882            ** Category:   FILUTL
2883            **
2884            ** Purpose:
2885            **      Copy a file from card into memory.
2886            **
2887            ** Entry:
2888            **      R3 = name of file to look for on card (zeroes if not
2889            **        specified).
2890            **      R2 = name of file to be used in RAM after it is read
2891            **        in (zeroes if not specified).
2892            **
2893            ** Exit:
2894            **      Returns if successful.
2895            **      R2[A]=pointer to file header of file just read in.
2896            **      If read fails, this code performs an error exit and
2897            **        does NOT return.
2898            **
2899            ** Calls:       ASRW4, CHKSUM, CLRALL, CMPALL, CMPWRT,
2900            **              CR??, CRDFAB, CRDOFF, CREATF, DONIBC, D1+13B,
2901            **              D1+29B, FILEF, FNDCLR, HDRHDR, IOAL36, LAKEYS,
2902            **              LC2TRK, MAKHDR, MEMCKL, MOVED3, NOCOMP, OFFSET,
2903            **              PLLCRD, R1DO37, R1TODO, RALIGN, RDSOC, RDYTRK,
2904            **              READ8S, READCS, READFL, RTODP, RWERR, SETBIT,
2905            **              SWPBYT, TRKDON, WRMSG, aslw5, asrw5, crlfnd,
2906            **              csrw5, fpoll, idiva, mvmem+, noscrl.
2907            **
2908            **              EXITS through BUFDAL.
2909            **
2910            ** Uses.......
2911            **              A,B,C,D,P,DO,D1,R0-R3,ST, SCRTCH, SNAPBF,
2912            **              8 or so levels in RSTKBF.
2913            **
2914            ** Stk lvls:   5
2915            **
2916            ** Detail:
2917            **      Creates an I/Obuffer for the card header and then
2918            **      creates the biggest possible file in the available
2919            **      memory.  Setting aside as many nibbles at the end as
2920            **      are necessary to maintain a tracks-read bitmap, reads
2921            **      the card into the file and then collapses the file to
2922            **      the proper size after the read.
2923            **
2924            **      Register usage in CRDFIL routine:
2925            **        R1[4-0]=address of data area in file (past header).
2926            **          [9-5]=amount of available memory in data area.
2927            **          [14-10]=size of bitmap.
2928            **        R2=pointer to header.
2929            **
2930            ** Algorithm:
```

```
2931            **      Perform MEMCHK on (size of file header) + (size of card
2932            **        header I/Obuffer) + (leeway); eMEM if failure.
2933            **      Compute remaining space (B=C-B).
2934            **      Add headersize for full file size (C=B+HDRSIZ).
2935            **      Create file {this creates the biggest allowable file,
2936            **        with R1 pointing at start of file header}.
2937            **      Write filename passed in R2 to file header.
2938            **      Zero out filetype field in file header.
2939            **      Hold address-of-file-data-area and size of file-
2940            **        without-header in R0[9-5] and R0[A], respectively.
2941            **      Determine size of bitmap needed { (#nibs in file data
2942            **        area)/(#nibs in four tracks) + 1 }.
2943            **      R1[A]=address of file data area {past header},
2944            **        R1[9-5]=(size of data area) - (size of bitmap)
2945            **          {eMEM if subtraction generates carry},
2946            **        R1[14-10]=size of bitmap {located at end of data
2947            **          area}.
2948            **      Clear all bits in bitmap.
2949            **      Create card header I/Obuffer.
2950            **      R2[A]=pointer to buffer area {past header}.
2951            **  1:  Send READ alignment message to display; CRDFAB, RTNABT
2952            **        if abort indicated by RALIGN.
2953            **      Send PULL CARD message to display.
2954            **      R4[S]=0 {indicate read has not occurred}.
2955            **      Read SOC and WPROT (RDSOC); goto 1 if error.
2956            **      Set FLGSRV.
2957            **      Read card header into card header I/Obuffer (MAKHDR);
2958            **        goto 1 if error.
2959            **      Read filename passed from file header.  If nonzero and
2960            **        doesn't match filename on card; eWRGNM and goto 1.
2961            **      {We have now determined the card header; hence
2962            **        filesize, name, etc.  There is no turning back.}
2963            **  2:  Set FLGSRV.
2964            **      Compute offset for this trk based on trk# (OFFSET).
2965            **      D[A]=#full {8-nibble} FIFO reads; D[S]=size of partial
2966            **        read * 2.
2967            **      If track will not fit in available memory, error out
2968            **        with eMEM.
2969            **  3:  If D=0 goto 4.
2970            **      Read 8 nibs from FIFO.
2971            **      Write at D0.
2972            **      Increment D0.
2973            **      Goto 3.
2974            **  4:  If there is no partial read goto 5.
2975            **      Perform partial read.
2976            **      Write at D0.
2977            **  5:  Turn off card reader.
2978            **      Compute checksum of data just read.
2979            **      Compare to checksum in header; if not match eRWERR and
2980            **        goto 6.
2981            **      Set bit corresponding to current trk# in bitmap.
2982            **      R4[S]="F" {indicate READ has occurred}.
2983            **  6:  If filename in file header = 0, copy name from card
2984            **        header I/Obuffer.
2985            **      Search for filename in file chain.
```

```
2986          **      If address found # address of this file then error out
2987          **        with eFEXST.
2988          **      If filetype in file header <> 0 goto 7.
2989          **      Compute filetype and security based on filetype in
2990          **        card header I/Obuffer (HDRHDR).
2991          **      If filetype unrecognized and not standard range then
2992          **        error out with eFTYPE.
2993          **      If filetype unrecognized and standard range and private
2994          **        then error out with eFPROT.
2995          **      Write unencoded filetype and flags to file header.
2996          **      Read filename from file header.
2997          **      If filename <> "keys    " then goto 7.
2998          **      If unencoded filetype <> =fKEY then error out with
2999          **        eFTYPE.
3000          **    7: Check if whole card set fits.  If not
3001          **        then error out with eMEM.
3002          **      Compute max trk#.
3003          **      Write max trk# to card header I/Obuffer.
3004          **      If R4[S]<>0 then send "Trk #xxx done" to display.
3005          **      Find first unread trk# (FNDCLR in bitmap).
3006          **      If next trk# > max trk# then goto 8.
3007          **      Send READ alignment message to display.
3008          **      If abort, deallocate file and exit through RTNABT.
3009          **      Read SOC and WPROT; goto 6 if error.
3010          **      { Now we will copy the card header to the card header
3011          **        I/Obuffer, selectively comparing nibbles as we go.
3012          **        If a read error occurs; goto 6.  If a compare error
3013          **        occurs, give eNOTST warning and goto 6.}
3014          **      Copy card header to card header I/Obuffer, comparing
3015          **        bytes 0, 5-26.5 {lonib of byte 26}.
3016          **      Compare header checksum with value on card.  Warn with
3017          **        eRWERR if not match and goto 6.
3018          **      Goto 2.
3019          **    8: { At this point, C[A] contains the length of the data
3020          **        area}.
3021          **      C[A]=C[A]+5 {compute file chain length}.
3022          **      If filetype <> LIF1 then goto 9.
3023          **      { File length on card is a multiple of one sector,
3024          **        which in general pads the LIF1 file a whole bunch.
3025          **        We seek to crunch the file down to its proper size.}
3026          **      Chain through LIF1 file looking for last record {FFFF}.
3027          **        If found {file not corrupt}, use smaller chain len.
3028          **      Stash implementation field in R3.]
3029          **      If copy code = 0, then implementation field[A] is the
3030          **        actual file length; add 5 for chain len.
3031          **      { We now have the file chain len--either from card
3032          **        header, looking at LIF1 chain or imp field}.
3033          **      Write file chain len to chain len field in file hdr.
3034          **      Compress file to proper length.
3035          **      If copy code=1, retrieve implementation field; insert
3036          **        into file after chain length; modify chain length.
3037          **      Send CR-LF to display.
3038          **      Return.
3039          **
3040          ** History:
```

```
3041                **
3042                **    Date     Programmer          Modification
3043                **  --------   ----------  ----------------------------------
3044                **  07/14/82   NM          Added documentation
3045                **  02/25/83   NM          Updated "CALLS" section
3046                **
3047                ************************************************************
3048                ************************************************************
3049 1D21D 737C =CRDFIL GOSUB  CR??           Rtn if cr present.
3050                * Check if room for file
3051 1D221 20          P=     0
3052 1D223 D2          C=0    A
3053 1D225 3147        LC(2)  (HDRSIZ)+2*36+7
3054 1D229 8F00        GOSBVL =MEMCKL        Room to create file & hdr bfr?
           000
3055 1D230 4E4         GOC    NOMEM          No.
3056                * Create available memory file
3057 1D233 D5          B=C    A
3058                *
3059 1D235 D2          C=0    A
3060 1D237 3152        LC(2)  HDRSIZ
3061 1D23B C9          C=B+C  A              Filesize with header
3062 1D23D 8F00        GOSBVL =CREATF        Create the file
           000
3063                * Abort routines from here on in must delete file!!!
3064                * Write out name
3065 1D244 767B        GOSUB  R1TODO         File pointer
3066 1D248 11A         C=R2                  Filename to use in RAM
3067 1D24B 1547        DAT0=C W              Write out to name field in hdr
3068 1D24F 16F         DO=DO+ 16
3069 1D252 D2          C=0    A
3070 1D254 144         DAT0=C A              Zero out 5 nibs at filetype
3071                * Partition avmem file into file and header areas
3072 1D257 16F         DO=DO+ 16
3073 1D25A 146         C=DAT0 A              File chain length
3074 1D25D D5          B=C    A              Hold
3075 1D25F 132         ADOEX                 Pointer to chain len
3076 1D262 20          P=     0
3077 1D264 D2          C=0    A
3078 1D266 305         LCHEX  5              Size of chain len field
3079 1D269 CA          A=A+C  A              Addr of start of file info
3080 1D26B E1          B=B-C  A              Remaining memory
3081 1D26D 7A9F        GOSUB  aslw5          A[9-5]=address
3082 1D271 D4          A=B    A              A[A]=(file-hdr)size
3083 1D273 100         R0=A                  Hold
3084                * Create bitmap
3085 1D276 CC          A=A-1  A
3086 1D278 5A0         GONC   CRDF13         Go if sufficient memory
3087 1D27B 7CA6  CRDFNM GOSUB  CRDFAB
3088 1D27F 67DC  NOMEM  GOTO   NROOM          Error out
3089 1D283 3405  CRDF13 LC(5)  2*4*(TRKSIZ)   #nibs in 4 tracks
           410
3090 1D28A 74CB        GOSUB  idiva          Determine bitmap size needed
3091 1D28E E4          A=A+1  A              # nibs for bitmap
3092 1D290 118         C=R0
```

```
3093 1D293 E2          C=C-A  A            Reduce avmem by bitmap size
3094 1D295 727F        GOSUB  aslw5
3095 1D299 DA          A=C    A
3096 1D29B 7C6F        GOSUB  aslw5
3097 1D29F 716F        GOSUB  csrw5
3098 1D2A3 DA          A=C    A            A[a]=addr,[9-5]=avmem,[14-10]=bmp
3099 1D2A5 101         R1=A
3100 1D2A8 42D         GOC    CRDFNM       Go if insufficient memory
3101 1D2AB 7466        GOSUB  CLRALL       Clear bitmap
3102             * Create i/obuffer for header
3103 1D2AF 7A3C        GOSUB  IOAL36       Allocate I/Obuffer
3104 1D2B3 137         CD1EX               We already know it fits
3105 1D2B6 10A         R2=C                Save pointer to buffer hdr
3106 1D2B9 137         CD1EX
3107             * Ready to prompt for first card
3108 1D2BC 7C8D CRDF20 GOSUB  RALIGN       "read: align, then endln"
3109 1D2C0 5A0         GONC   CRDF30       No abort
3110 1D2C3 7466 RDABT! GOSUB  CRDFAB       Abort
3111 1D2C7 61FB        GOTO   RTNABT
3112             * Pull card
3113 1D2CB 787C CRDF30 GOSUB  PLLCRD
3114 1D2CF AC2         C=0    S
3115 1D2D2 10C         R4=C                Indicate read hasn't occurred
3116 1D2D5 779B        GOSUB  RDSOC        Get past start of card
3117 1D2D9 42E         GOC    CRDF20       Error
3118 1D2DC 7ADD        GOSUB  READFL       Set flgsrv
3119 1D2E0 11A         C=R2
3120 1D2E3 134         D0=C                Point at i/obuffer
3121 1D2E6 7676        GOSUB  MAKHDR       Read card header into i/obuffer
3122 1D2EA 41D         GOC    CRDF20       Go if error
3123 1D2ED 77CA        GOSUB  RTODP
3124 1D2F1 70FB        GOSUB  D1+13B
3125 1D2F5 1577        C=DAT1 W            Read filename
3126 1D2F9 113         A=R3                Expected name
3127 1D2FC 978         ?A=0   W            Was anything expected?
3128 1D2FF 31          GOYES  CRD130       No
3129 1D301 972         ?A=C   W            Yes.  match?
3130 1D304 E0          GOYES  CRD130       Yes
3131 1D306 1D0         D1=(2) =eWRGNM      "wrong name"
3132 1D30A 7AAC        GOSUB  WRMSG
3133 1D30E 6DAF        GOTO   CRDF20       Try again
3134             * Header is now determined
3135             * Read loop starts here (in the middle of the card, of course.)
3136 1D312 74AD CRD130 GOSUB  READFL       Set flgsrv
3137 1D316 77E6        GOSUB  OFFSET       Compute card starting offset
3138 1D31A D8          B=A    A            Stash
3139 1D31C CA          A=A+C  A            Add trksize for ending offset
3140 1D31E AF3         D=0    W
3141 1D321 D7          D=C    A
3142 1D323 C7          D=D+D  A
3143 1D325 817         DSRC                Read counter
3144 1D328 119         C=R1
3145 1D32B C9          C=C+B  A            Starting address
3146 1D32D 134         D0=C
3147 1D330 70DE        GOSUB  csrw5        Avail mem
```

```
3148 1D334 8BA          ?A<=C  A          Room for this card?
3149 1D337 60           GOYES  CRD140     Yes
3150 1D339 614F         GOTO   CRDFNM     No room for card set
3151 1D33D CF    CRD140 D=D-1  A          Any more full words to read?
3152 1D33F 441          GOC    CRD150     No
3153 1D342 769D         GOSUB  READ8S     Yes, read.
3154 1D346 4D1          GOC    CRD155     Go if error or abort.
3155 1D349 1587         DATO=A 8          Write it out
3156 1D34D 167          D0=D0+ 8
3157 1D350 6CEF         GOTO   CRD140
3158 1D354 81F   CRD150 DSRB               Size of partial read in nibs
3159 1D357 A4F          D=D-1  S          0?
3160 1D35A ACB          C=D    S
3161 1D35D 441          GOC    CRD160     Yes
3162 1D360 729D         GOSUB  READCS     No.  do partial read.
3163 1D364 404   CRD155 GOC    CRDERR
3164 1D367 ACB          C=D    S
3165 1D36A 80DF         P=C    15         #nibs in read
3166 1D36E 1501         DATO=A WP         Write out partial read
3167 1D372 7D3D  CRD160 GOSUB  CRDOFF     Turn off card reader
3168 1D376 7786         GOSUB  OFFSET     Trk offset to a, trksize to c
3169 1D37A 134          D0=C              Hold size
3170 1D37D 119          C=R1              Pointer to start of data
3171 1D380 C2           C=A+C  A          Addr of this track
3172 1D382 136          CD0EX             Adr in d0, size in c
3173 1D385 756A         GOSUB  CHKSUM     Compute checksum
3174 1D389 11A          C=R2
3175 1D38C 135          D1=C
3176 1D38F 794B         GOSUB  D1+29B
3177 1D393 175          D1=D1+ 2*3
3178 1D396 15F3         C=DAT1 4          Expected checksum
3179 1D39A 23           P=     3
3180 1D39C 912          ?A=C   WP         Match?
3181 1D39F A0           GOYES  CRD180     Yes, read was good.
3182 1D3A1 7AAB         GOSUB  RWERR      No.  "checksum error"
3183 1D3A5       CRDERR
3184 1D3A5 6910         GOTO   CRDNXT     On to next card.
3185 1D3A9       CRD180
3186 1D3A9 7B0A         GOSUB  RTODP
3187 1D3AD 171          D1=D1+ 2
3188 1D3B0 14F          C=DAT1 B          Current trk#
3189 1D3B3 7D96         GOSUB  SETBIT     Mark as read
3190 1D3B7 2F           P=     15
3191 1D3B9 30F          LCHEX  F
3192 1D3BC 10C          R4=C              Indicate that read has occurred
3193 1D3BF 7522  CRDNXT GOSUB  R1D037     Point at filename in file hdr
3194 1D3C3 1527         A=DATO W
3195 1D3C7 97C          ?A#0   W          Filename specified?
3196 1D3CA 41           GOYES  CRDN10     Yes
3197 1D3CC 11A          C=R2
3198 1D3CF 135          D1=C
3199 1D3D2 7F0B         GOSUB  D1+13B     Point at filename
3200 1D3D6 1537         A=DAT1 W          Read filename from card
3201 1D3DA 1507         DATO=A W          Write to file header
3202 1D3DE 20    CRDN10 P=     0
```

```
3203 1D3E0 8F00          GOSBVL =FILEF          Search for file
           000
3204 1D3E7 4D1           GOC    CRDN20          Go if doesn't exist
3205 1D3EA 7AF1          GOSUB  R1D037
3206 1D3EE 132           ADOEX                  Pointer to current file
3207 1D3F1 137           CD1EX                  Pointer to found file
3208 1D3F4 8A2           ?A=C   A               Same?
3209 1D3F7 E0            GOYES  CRDN20          Yes.  not duplicate name.
3210 1D3F9 7E25          GOSUB  CRDFAB
3211 1D3FD 1D00          D1=(2) =eFEXST
3212 1D401 673C          GOTO   errout          "file exists"
3213 1D405 119   CRDN20  C=R1
3214 1D408 135           D1=C
3215 1D40B 1CF           D1=D1- 16
3216 1D40E 1C4           D1=D1- 5               Point at filetype in file hdr
3217 1D411 147           C=DAT1 A
3218 1D414 8AE           ?C#0   A               Have we done filetype yet?
3219 1D417 56            GOYES  CRHD70          Yes
3220 1D419 11A           C=R2                   No.
3221 1D41C 134           D0=C                   Point at start of card hdr.
3222 1D41F 7F36          GOSUB  HDRHDR          Get unencoded filetype and flags
3223 1D423 5A0           GONC   CRHD10          Go if recognized or std range
3224 1D426 7105          GOSUB  CRDFAB          Unrecognized.  collapse file.
3225 1D42A 643B          GOTO   FTYPER          Error out
3226 1D42E 831   CRHD10  ?XM=0                  Unrecognized & std & private?
3227 1D431 E0            GOYES   CRHD60         No
3228 1D433 74F4          GOSUB  CRDFAB          Yes.  abort.
3229 1D437 1D00          D1=(2) =eFPROT         File protection error
3230 1D43B 6DFB          GOTO   errout          Error out
3231 1D43F 1593  CRHD60  DAT1=A 4               Write out unencoded filetype
3232 1D443 173           D1=D1+ 4
3233 1D446 14D           DAT1=C B               Write out flags
3234 1D449 D8            B=A    A               Stash filetype
3235 1D44B 1C3           D1=D1- 4
3236 1D44E 1CF           D1=D1- 16
3237 1D451 20            P=     0
3238 1D453 8F00          GOSBVL =LAKEYS         A=C="    syek"
           000
3239 1D45A 1537          A=DAT1 W               Read filename from file header
3240 1D45E 976           ?A#C   W               Name = "keys    "?
3241 1D461 B1            GOYES  CRHD70          No
3242 1D463 33C0          LC(4)  =fKEY           Yes
           2E
3243 1D469 23            P=     3
3244 1D46B 911           ?B=C   WP              Are we reading in a key file?
3245 1D46E E0            GOYES  CRHD70          Yes
3246 1D470 77B4          GOSUB  CRDFAB          No.  Abort
3247 1D474 6AEA          GOTO   FTYPER
3248 1D478 620E  crdfnm  GOTO   CRDFNM
3249 1D47C 11A   CRHD70  C=R2
3250 1D47F 134           D0=C
3251 1D482 169           D0=D0+ 2*5
3252 1D485 71C5          GOSUB  D0NIBC          Filesize in nibs
3253 1D489 185           D0=D0- 2*3             Point at max trk#
3254 1D48C 111           A=R1
```

```
3255 1D48F 7F7D          GOSUB  asrw5          Available space
3256 1D493 136           CDOEX
3257 1D496 167           DO=DO+ 8              8 nibbles extra in case file...
3258 1D499 136           CDOEX                 has implementation field
3259 1D49C 8B2           ?A<C   A              Space>=filesize?
3260 1D49F 9D            GOYES  crdfnm         No.  Insufficient memory.
3261             * Recalculate trksize in case read of wrong set munched it
3262 1D4A1 137           CD1EX
3263 1D4A4 1C7           D1=D1- 8              Actual size of card set
3264 1D4A7 133           AD1EX
3265 1D4AA 7DE6          GOSUB  LC2TRK         Size of a track
3266 1D4AE 70A9          GOSUB  idiva          Filesize/trksize=#trks
3267 1D4B2 8A9           ?B=0   A              Any remainder?
3268 1D4B5 50            GOYES  CRD220         No.
3269 1D4B8 B64           A=A+1  B              Yes.  Inc #trks.
3270 1D4BA 148   CRD220  DAT0=A B              Write out max trk#
3271             *
3272 1D4BD 11C           C=R4
3273 1D4C0 94A           ?C=0   S              Was a read done?
3274 1D4C3 C0            GOYES  CRD222         No
3275 1D4C5 726A          GOSUB  fpoll          Poll for card read processing
3276 1D4C9 43            CON(2) =pRCRD
3277 1D4CB 7DDA          GOSUB  TRKDON         "trk #### done"
3278 1D4CF D2    CRD222  C=0    A
3279 1D4D1 E6            C=C+1  A
3280 1D4D3 7816          GOSUB  FNDCLR         Find first unread trk#
3281 1D4D7 471           GOC    CRD225         Hmmm... guess we're done
3282 1D4DA 7AD8          GOSUB  RTODP
3283 1D4DE 173           D1=D1+ 2*2
3284 1D4E1 14B           A=DAT1 B              Read max trk#
3285 1D4E4 1C1           D1=D1- 2
3286 1D4E7 14D           DAT1=C B              Write out next trk#
3287 1D4EA 9EE           ?C<=A  B              Are we done?
3288 1D4ED 91            GOYES  CRD240         No
3289 1D4EF 6D01  CRD225  GOTO   CRDDON         Tidy up
3290 1D4F3 743A  CRHD35  GOSUB  fpoll          ABORT
3291 1D4F7 33            CON(2) =pCRDAB
3292 1D4F9 831           ?XM=0                 Handled?
3293 1D4FC 60            GOYES  CRD230         Yes.  Handler collapsed, etc.
3294 1D4FE 64CD          GOTO   RDABT!         No.  Collapse and abort.
3295 1D502 66B9  CRD230  GOTO   RTNABT         Just abort.
3296 1D506 724B  CRD240  GOSUB  RALIGN         Prompt for new card
3297 1D50A 48E           GOC    CRHD35         Done.  time to pack.
3298 1D50D 765A          GOSUB  RDYTRK         Pull card.
3299 1D511 AC2           C=0    S
3300 1D514 10C           R4=C                  Indicate read hasn't occurred
3301 1D517 7559          GOSUB  RDSOC          Get past soc stuff
3302 1D51B 4B2           GOC    CRDER?         Go if error
3303 1D51E 789B          GOSUB  READFL         Set flgsrv for header field
3304 1D522 11A           C=R2
3305 1D525 134           DO=C                  Point at header area
3306 1D528 AF3           D=0    W              Space for checksum
3307 1D52B 7DAB          GOSUB  READ8S         Read bytes 0-3.
3308 1D52F 471           GOC    CRDER?
3309 1D532 AF2           C=0    W
```

```
3310 1D535 A6E          C=C-1  B
3311 1D538 75F5         GOSUB  CMPWRT        Copy to ram.  compare subfmt.
3312 1D53C 5E0          GONC   CRD270        Go if compared ok
3313 1D53F 1D00 BADHDR  D1=(2) =eNOTIN
3314 1D543 717A         GOSUB  WRMSG         "card not in set"
3315 1D547 677E CRDER?  GOTO   CRDNXT
3316 1D54B 7D8B CRD270  GOSUB  READ8S        Read bytes 4-7 of header
3317 1D54F 47F          GOC    CRDER?
3318 1D552 AF2          C=0    W
3319 1D555 A7E          C=C-1  W
3320 1D558 AE2          C=0    B
3321 1D55B 72D5         GOSUB  CMPWRT        Copy.  compare bytes 5-7.
3322 1D55F 4FD          GOC    BADHDR        Go if compare error.
3323 1D562 2F           P=     15
3324 1D564 303          LCHEX  3
3325 1D567 AC7          D=C    S             Prepare for 4 full compares
3326 1D56A 7E6B CRD280  GOSUB  READ8S        Read next 4 bytes from card
3327 1D56E 48D          GOC    CRDER?
3328 1D571 76B5         GOSUB  CMPALL        Write and compare
3329 1D575 49C          GOC    BADHDR
3330 1D578 A4F          D=D-1  S
3331 1D57B 5EE          GONC   CRD280
3332 1D57E 7A5B         GOSUB  READ8S        Read bytes 24-27
3333 1D582 44C  CRD285  GOC    CRDER?
3334 1D585 AF2          C=0    W
3335 1D588 26           P=     6
3336 1D58A A1E          C=C-1  WP
3337 1D58D 70A5         GOSUB  CMPWRT        Compare bytes 24-26.5 (pss lonib)
3338 1D591 4DA          GOC    BADHDR
3339 1D594 744B         GOSUB  READ8S        Read bytes 28-31
3340 1D598 49E          GOC    CRD285
3341 1D59B 78A5         GOSUB  NOCOMP        Copy... don't compare
3342 1D59F 793B         GOSUB  READ8S        Read bytes 32-35
3343 1D5A3 4ED          GOC    CRD285
3344 1D5A6 1587         DAT0=A 8             Write out
3345 1D5AA 23           P=     3
3346 1D5AC A9B          C=D    WP
3347 1D5AF A12          C=A+C  WP            Add data cksum to header
3348 1D5B2 550          GONC   CRD290
3349 1D5B5 B16          C=C+1  WP            Wraparound carry
3350 1D5B8 BF7  CRD290  DSR    W
3351 1D5BB BF7          DSR    W
3352 1D5BE BF7          DSR    W
3353 1D5C1 F7           DSR    A
3354 1D5C3 A1B          C=C+D  WP            Compress whole checksum
3355 1D5C6 550          GONC   CRD300
3356 1D5C9 B16          C=C+1  WP
3357 1D5CC D7   CRD300  D=C    A
3358 1D5CE F6           CSR    A
3359 1D5D0 F6           CSR    A
3360 1D5D2 CB           C=C+D  A             Compress to 1 byte w/o wraparound
3361 1D5D4 8F00         GOSBVL =ASRW4        Align expected cksum
           000
3362 1D5DB 962          ?A=C   B             Agreement?
3363 1D5DE B1           GOYES  CRD310        Yes
```

```
3364 1D5E0 7B69         GOSUB  RWERR          No.  bad header checksum.
3365 1D5E4 6ADD         GOTO   CRDNXT
3366 1D5E8 119   R1D037 C=R1
3367 1D5EB 134         DO=C
3368 1D5EE 184         DO=DO- 5
3369 1D5F1 18F         DO=DO- 16
3370 1D5F4 18F         DO=DO- 16
3371 1D5F7 01          RTN
3372 1D5F9 681D  CRD310 GOTO   CRD130         Read data field
3373 1D5FD 112   CRDDON A=R2
3374 1D600 131         D1=A
3375 1D603 179         D1=D1+ 2*5            Point at file length
3376 1D606 D2          C=0    A
3377 1D608 15F3        C=DAT1 4
3378 1D60C C6          C=C+C  A              File len in nibs
3379 1D60E 134         DO=C
3380 1D611 164         DO=DO+ 5              File chain length
3381 1D614 11A         C=R2
3382 1D617 136         CDOEX                 Point at card header
3383 1D61A 111         A=R1
3384 1D61D 131         D1=A                  Point at file
3385 1D620 1CF         D1=D1- 16
3386 1D623 1C4         D1=D1- 5
3387 1D626 143         A=DAT1 A              Read filetype
3388 1D629 CC          A=A-1  A
3389 1D62B 23          P=     3
3390 1D62D 91C         ?A#0   WP             Lif1?
3391 1D630 E4          GOYES  CRD400         No
3392 1D632 17F         D1=D1+ 16             Yes
3393 1D635 174         D1=D1+ 5              Point at data
3394 1D638 D7          D=C    A              Stash current chain len
3395 1D63A D2          C=0    A
3396 1D63C 20          P=     0
3397 1D63E 305         LCHEX  5              Start of new chain len comp
3398 1D641 D5          B=C    A              Will compute in B
3399 1D643 23          P=     3
3400 1D645 15B3  CRD320 A=DAT1 4             Reclen in bytes
3401 1D649 173         D1=D1+ 4
3402 1D64C 8E00         GOSUBL =SWPBYT        Put reclen in right order
           00
3403 1D652 F0          ASL    A
3404 1D654 F4          ASR    A              Clear upper nibble
3405 1D656 E5          B=B+1  A
3406 1D658 E5          B=B+1  A
3407 1D65A E5          B=B+1  A
3408 1D65C E5          B=B+1  A
3409 1D65E D9          C=B    A              In case next step carries
3410 1D660 B14         A=A+1  WP             Reclen=ffff?
3411 1D663 4A1         GOC    CRD400         Yes.  we have new chain len
3412 1D666 81C         ASRB
3413 1D669 C4          A=A+A  A              Reclen if even, reclen+1 if odd
3414 1D66B C4          A=A+A  A              Convert reclen to nibs
3415 1D66D C8          B=A+B  A              Add recsize to chain len
3416 1D66F 137         CD1EX
3417 1D672 C2          C=A+C  A              Point past record
```

```
3418 1D674 137          CD1EX
3419 1D677 8B3          ?C<D   A            Exceeded original size?
3420 1D67A BC           GOYES  CRD320       No
3421 1D67C DB    CRD390 C=D    A            Restore original chain len
3422 1D67E       CRD400
3423 1D67E 16F          D0=D0+ 16
3424 1D681 16F          D0=D0+ 16
3425 1D684 169          D0=D0+ 2*5
3426 1D687 15A7         A=DAT0 8
3427 1D68B 103          R3=A                Save implementation field
3428 1D68E D8           B=A    A            May be file length in nibs
3429 1D690 111          A=R1
3430 1D693 131          D1=A                Point at file
3431 1D696 1CF          D1=D1- 16           Point at copy code
3432 1D699 1532         A=DAT1 XS
3433 1D69D A2C          A=A-1  XS           Copy code=0?
3434 1D6A0 5D0          GONC   CRD410       No
3435 1D6A3 D9           C=B    A            Yes.  Imp field is len in nibs
3436 1D6A5 137          CD1EX
3437 1D6A8 174          D1=D1+ 5            Add 5 to get file chain len
3438 1D6AB 137          CD1EX
3439 1D6AE 92C   CRD410 ?A#0   XS           Copy code=1?
3440 1D6B1 92           GOYES  CRD420       No.
3441             *
3442 1D6B3 D7           D=C    A            Hold block length
3443 1D6B5 135          D1=C
3444 1D6B8 177          D1=D1+ 8            New chain length
3445 1D6BB 119          C=R1                Start of source
3446 1D6BE CB           C=C+D  A
3447 1D6C0 134          D0=C                End source
3448 1D6C3 137          CD1EX               New chain len to C
3449 1D6C6 177          D1=D1+ 8            End dest
3450 1D6C9 DF           CDEX   A            New ch len to D, blk len to C
3451 1D6CB 8E00         GOSUBL =MOVED3      Make room for impl field
          00
3452             *
3453 1D6D1 11B          C=R3                Get impl. field
3454 1D6D4 15C7         DAT0=C 8            Write it out
3455 1D6D8 DB           C=D    A            Recall chain length
3456 1D6DA 111   CRD420 A=R1
3457 1D6DD 131          D1=A
3458 1D6E0 1C4          D1=D1- 5            Point at current chain len
3459 1D6E3 143          A=DAT1 A            Chain len of unshrunk file
3460 1D6E6 E2           C=C-A  A            Subtract from (smaller) chnlen
3461 1D6E8 D5           B=C    A            Hold for MVMEM+
3462 1D6EA 137          CD1EX
3463 1D6ED CA           A=A+C  A            Add to chnlen for end of file
3464 1D6EF 137          CD1EX
3465 1D6F2 1CF          D1=D1- 16
3466 1D6F5 1CF          D1=D1- 16           Start of this file
3467 1D6F8 137          CD1EX
3468 1D6FB 10A          R2=C                Hold pointer to file
3469 1D6FE 7552         GOSUB  MVMEM+       Shrink file
3470 1D702 8C18         GOLONG FINCRD       Done
          3F
```

```
   3471 1D708       space2 BSS    3              Extra space from packing
```

```
3472                    STITLE PROTECT and UNPROTECT
3473                ****************************************************************
3474                ****************************************************************
3475                **
3476                ** Name:     PROTCT  -  Protect/Unprotect Card
3477                ** Name:     UNPROT  -  Protect/Unprotect Card
3478                **
3479                ** Category:   STEXEC
3480                **
3481                ** Purpose:
3482                **      Write-protect or write-unprotect a card.
3483                **
3484                ** Entry:
3485                **      D0 = PC.
3486                **      Jumped on PROTECT or UNPROTEC token.
3487                **
3488                ** Exit:
3489                **      Through NXTSTM.
3490                **
3491                ** Detail:
3492                **      PROTECT.
3493                **      UNPROTECT.
3494                **
3495                ** Algorithm:
3496                **   PROTCT: C=FFFFFFFFFFFFFFF.
3497                **           Goto 1.
3498                **   UNPROT: C=0000000000000000.
3499                **        1: R1=C.
3500                **        2: Check for card reader presence; error out with
3501                **             eDVCNF if not there {CR??}.
3502                **           If R1[S]=0 then send "UNPR" alignment message to
3503                **             display, else send "PROT" alignment message to
3504                **             display.
3505                **           If abort, exit through RTNABT.
3506                **           Send "pull card" to display.
3507                **           GETSOC; goto 2 if error or compare failure.
3508                **           GETSIZ; goto 2 if error or compare failure.
3509                **           Set write mode.
3510                **           Write 16 nibbles of zeroes.
3511                **           Write 8 nibbles of R1.
3512                **           Write 2 nibbles of R1.
3513                **           Wait until FIFO empty.
3514                **           Turn off card reader.
3515                **           Exit through NXTSTM.
3516                **
3517                ** History:
3518                **
3519                **    Date     Programmer             Modification
3520                ** --------   ----------   ---------------------------------
3521                ** 06/25/82   NM           Added documentation
3522                **
3523                ****************************************************************
3524                ****************************************************************
3525 1D70B 0000          REL(5) =STOPDC
          0
```

```
3526 1D710 0000          REL(5) =RTNCC
           0
3527 1D715 AF2  =PROTCT C=0    W
3528 1D718 A7E          C=C-1  W
3529 1D71B 4F0          GOC    PRTU05          B.E.T.
3530               *
3531 1D71E 0000          REL(5) =UNPRDC
           0
3532 1D723 0000          REL(5) =UNPRTP
           0
3533 1D728 AF2  =UNPROT C=0    W
3534 1D72B 109   PRTU05 R1=C                   Store protect/unprotect field
3535 1D72E 8E06  PRTUNP GOSUBL CR??
           7F
3536 1D734 119          C=R1
3537 1D737 1D00         D1=(2) =eUALGN         UNPROTECT prompt
3538 1D73B 94A          ?C=0   S               Protect or unprotect?
3539 1D73E 50           GOYES  PRMPT           Unprotect
3540 1D740 1C0          D1=D1- 1               PROTECT prompt
3541 1D743 7909  PRMPT  GOSUB  ALIGN
3542 1D747 5F0          GONC   PRTU10
3543              *!!
3544              *!! Replace following two lines with:
3545              *!!
3546              *!! PRTABT GOSUB  CRDOFF
3547              *!!
3548 1D74A 85E   PRTABT ST=1   14              Set don't-continue flag
3549 1D74D 7269  PRTEXT GOSUB  CRDOFF
3550 1D751 8C96         GOLONG RTNAB+
           7F
3551 1D757 8EAE  PRTU10 GOSUBL PLLCRD          "pull card"
           7F
3552 1D75D 8E36         GOSUBL GETSOC          Get soc field from card
           6F
3553 1D763 4AC   PROTAB GOC    PRTUNP          Error.  Try again
3554 1D766 912          ?A=C   WP              Found soc?
3555 1D769 C0           GOYES  PRTU30          Yes
3556 1D76B 8EEC  PRTU25 GOSUBL UNKCD           No
           7F
3557 1D771 6CBF         GOTO   PRTUNP          Prompt again
3558 1D775 8E36  PRTU30 GOSUBL GETSIZ          Read card size field
           6F
3559 1D77B 47E          GOC    PROTAB
3560 1D77E 916          ?A#C   WP              Right size?
3561 1D781 AE           GOYES  PRTU25          No.
3562 1D783 7709         GOSUB  WRITE           Get into write mode
3563 1D787 7B6A         GOSUB  WRT2-0          Write 16 zeroes
3564 1D78B 47D          GOC    PROTAB
3565 1D78E 7F29         GOSUB  WRITFL          Set flgsrv
3566 1D792 40D          GOC    PROTAB
3567 1D795 111          A=R1
3568 1D798 7D49         GOSUB  WRIT8S          Write protect/unprotect field
3569 1D79C 46C          GOC    PROTAB
3570 1D79F 7069         GOSUB  WRIT4S          Write two more bytes
3571 1D7A3 4FB          GOC    PROTAB
```

```
3572 1D7A6 7E89          GOSUB  WAITM+
3573 1D7AA 48B           GOC    PROTAB
3574              *!!
3575              *!! Replace following two lines with:
3576              *!!
3577              *!!         GOSUB  CRDOFF
3578              *!!         GOLONG RTNAB+
3579              *!! space1 NIBHEX 00
3580              *!!
3581 1D7AD 5F9           GONC   PRTEXT          Done
3582 1D7B0 0000  space1 NIBHEX 000000          Extra space from packing
           00
```

```
3583                    STITLE CAT CARD
3584                ********************************************************************
3585                ********************************************************************
3586                **
3587                ** Name:    CATCRD  -  CAT CARD
3588                **
3589                ** Category:   STEXEC
3590                **
3591                ** Purpose:
3592                **      Catalog the contents of a card.
3593                **
3594                ** Entry:
3595                **      Called from CAT routine.
3596                **
3597                ** Exit:
3598                **      Through NXTSTM.
3599                **
3600                ** Detail:
3601                **      CAT CARD.
3602                **
3603                ** Algorithm:
3604                **      Check for card reader present.  Error out with eDVCNF
3605                **        if not found.
3606                **      Create I/Obuffer to contain card header and file
3607                **        header; eMEM if not room.
3608                **    1: Send "CAT" alignment message to display.
3609                **      RDSOC; goto 1 if error.
3610                **      Set FLGSRV.
3611                **      Read header into I/Obuffer {MAKHDR}; goto 1 if error.
3612                **      Turn off card reader.
3613                **      Send name to display.
3614                **      Send eTRKOF {"(trk <trk#> of <#trks>)"}, using trk# and
3615                **        maxtrk# from card header I/Obuffer.  {will append
3616                **        to name already sent}
3617                **      Create dummy file header in I/Obuffer immediately after
3618                **        card header buffer.  Copy name, filetype, flags.
3619                **        Convert file length (in bytes, from card hdr) into
3620                **        a file chain length and write out.  Convert creation
3621                **        date {given as seconds in century} to YYMMDDHHMM and
3622                **        write out.
3623                **      Call CAT$20 to prepare catalog entry.
3624                **      Call CAT100 to send to display.
3625                **      Deallocate I/Obuffer.
3626                **      S14=0 {clear don't continue flag}.
3627                **      Exit through NXTSTM.
3628                **
3629                ** History:
3630                **
3631                **    Date     Programmer              Modification
3632                **  --------   ----------   ---------------------------------
3633                **  06/25/82   NM           Added documentation
3634                **
3635                ********************************************************************
3636                ********************************************************************
3637 1D7B6 842   =CT$CRD ST=0   2              Indicate this is CAT$
```

```
3638 1D7B9 8EAE          GOSUBL CRPRS?          CR present?
           6F
3639 1D7BF 511           GONC   CATC05          Yes.
3640 1D7C2 8C9D          GOLONG CR??10          No.  Device not found.
           6F
3641 1D7C8 8E6C =CATCRD  GOSUBL CR??            Send CR and check for CR present.
           6F
3642 1D7CE 852           ST=1   2               Indicate this is CAT
3643 1D7D1 20    CATC05  P=     0
3644 1D7D3 D2            C=0    A
3645 1D7D5 31D6          LC(2)  2*36+37         Size of card hdr + file hdr
3646 1D7D9 8E61          GOSUBL IOAL+           Allocate I/O buffer
           7F
3647 1D7DF 460           GOC    CATC10          Go if fit
3648 1D7E2 6C9A          GOTO   NOMEM
3649 1D7E6 137   CATC10  CD1EX
3650 1D7E9 10A           R2=C                   Save pointer to buffer
3651 1D7EC 1D00  CATC20  D1=(2) =eCALGN
3652 1D7F0 7C58          GOSUB  ALIGN           "cat: align, then rtn"
3653 1D7F4 560           GONC   CATC30
3654 1D7F7 625F          GOTO   PRTABT          RTNABT
3655 1D7FB 8E64  CATC30  GOSUBL PLLCRD          "pull card"
           7F
3656 1D801 8E96          GOSUBL RDSOC           Get past soc
           6F
3657 1D807 44E           GOC    CATC20          Go if error
3658 1D80A 7CA8          GOSUB  READFL          Set flgsrv for header field
3659 1D80E 11A           C=R2
3660 1D811 134           DO=C                   Point at header buffer
3661 1D814 7841          GOSUB  MAKHDR          Read in header
3662 1D818 43D           GOC    CATC20          Go if error
3663 1D81B 7498          GOSUB  CRDOFF          Turn off card reader
3664 1D81F 862           ?ST=0  2               Is this CAT$?
3665 1D822 24            GOYES  CAT$C1          Yes.  Skip message.
3666 1D824 11A           C=R2
3667 1D827 134           DO=C
3668 1D82A 16C           DO=DO+ 13
3669 1D82D 16C           DO=DO+ 13
3670 1D830 1567          C=DATO W               Read name
3671 1D834 10B           R3=C                   Stash
3672 1D837 11B   CATC40  C=R3                   Loop to write out name
3673 1D83A 96A           ?C=0   B               Done?
3674 1D83D 91            GOYES  CATC50          Yes
3675 1D83F AEA           A=C    B               No.  next char.
3676 1D842 BF6           CSR    W
3677 1D845 BF6           CSR    W
3678 1D848 10B           R3=C
3679 1D84B 8F00          GOSBVL =DSPCHA         Display current char
           000
3680 1D852 64EF          GOTO   CATC40
3681 1D856 1A00  CATC50  DO=(4) =eTRKOF
           00
3682 1D85C 26            P=     6               Normal msg with delay
3683 1D85E 8EB0          GOSUBL RDYT10          " (trk ### of ###)"
           7F
```

```
3684 1D864 11A   CAT$C1 C=R2
3685 1D867 134          DO=C
3686 1D86A 74F1         GOSUB  HDRHDR          Compute unencoded ftype & flags
3687 1D86E AE5          B=C    B               Hold flags
3688 1D871 11A          C=R2
3689 1D874 134          DO=C                   Point to card header
3690 1D877 1F85         D1=(5) 2*36+16
           000
3691 1D87E 133          AD1EX
3692 1D881 CA           A=A+C  A
3693 1D883 133          AD1EX                  Point to file header + 16
3694 1D886 141          DAT1=A A               Write out filetype
3695 1D889 173          D1=D1+ 4
3696 1D88C AE9          C=B    B
3697 1D88F 14D          DAT1=C B               Write out flags
3698 1D892 169          DO=DO+ 2*5
3699 1D895 71B1         GOSUB  DONIBC          Filesize in nibs
3700 1D899 136          CDOEX
3701 1D89C 164          DO=DO+ 5
3702 1D89F 136          CDOEX                  File chain length
3703 1D8A2 17B          D1=D1+ 12
3704 1D8A5 145          DAT1=C A               Write to file header
3705 1D8A8 167          DO=DO+ 2*4
3706 1D8AB AF2          C=0    W
3707 1D8AE 15E7         C=DAT0 8               Read creation date
3708 1D8B2 8F00         GOSBVL =YMDHO1         To ymdhms
           000
3709 1D8B9 BF6          CSR    W
3710 1D8BC BF6          CSR    W
3711 1D8BF 1C9          D1=D1- 10
3712 1D8C2 15D9         DAT1=C 10              Write out to file header
3713 1D8C6 167          DO=DO+ 2*4
3714 1D8C9 1567         C=DAT0 W               Get filename
3715 1D8CD 1C5          D1=D1- 6
3716 1D8D0 1CF          D1=D1- 16
3717 1D8D3 1557         DAT1=C W               Write out name
3718 1D8D7 840          ST=0   0
3719 1D8DA 20           P=     0
3720 1D8DC 8F00         GOSBVL =CAT$20         Prepare catalog entry
           000
3721               *!!
3722               *!! Insert here:
3723               *!!
3724               *!!        GOSUBL BUFDAL
3725               *!!
3726 1D8E3 862          ?ST=0  2               Is this CAT$
3727 1D8E6 D1           GOYES  CAT$C2          Yes.
3728 1D8E8 8F00         GOSBVL =OBCOLL         Collapse output buffer.
           000
3729 1D8EF 8F00         GOSBVL =CAT100         Send (AVMEMS) to display
           000
3730 1D8F6 8F00         GOSBVL =DSPDLY
           000
3731 1D8FD 8C5C         GOLONG NEXTST          Off to scroll
           5F
```

```
3732                *!!
3733                *!! Replace following two lines with:
3734                *!!
3735                *!! CAT$C2 ST=1   0
3736                *!!
3737 1D903 8EAF     CAT$C2 GOSUBL BUFDAL
           5F
3738 1D909 850             ST=1   0
3739 1D90C 8D00            GOVLNG =CAT$66
           000
```

```
3740                    STITLE Copy card utilities
3741                ********************************************************************
3742                ********************************************************************
3743                **
3744                ** Name:    CLRALL  -  Clear Bitmap
3745                **
3746                ** Category:   LOCAL
3747                **
3748                ** Purpose:
3749                **      Clear tracks-read bitmap prior to card read operation.
3750                **
3751                ** Entry:
3752                **      R1[4-0]=address of data area,
3753                **        [9-5]=size of data area,
3754                **        [14-10]=size of bitmap.
3755                **
3756                ** Exit:
3757                **      Through WIPOUT.
3758                **
3759                ** Calls:      CSRW5, WIPOUT (falls through).
3760                **
3761                ** Uses.......
3762                **             A,C,P,D0,D1.
3763                **
3764                ** Stk lvls:   1
3765                **
3766                ** Algorithm:
3767                **      Compute location of bitmap (R1[A] + R1[9-5]) and
3768                **        location (R1[14-10]).
3769                **      WIPOUT.
3770                **
3771                ** History:
3772                **
3773                **    Date     Programmer              Modification
3774                **  --------   ----------   ---------------------------------
3775                **  06/25/82   NM           Added documentation
3776                **
3777                ********************************************************************
3778                ********************************************************************
3779 1D913 119   CLRALL C=R1
3780 1D916 DA          A=C    A               File data address
3781 1D918 78E8        GOSUB  csrw5
3782 1D91C CA          A=A+C  A
3783 1D91E 131         D1=A                   Point at bitmap
3784 1D921 7FD8        GOSUB  csrw5           Size of bitmap
3785 1D925 8C00        GOLONG =WIPOUT
          00
3786                ********************************************************************
3787                ********************************************************************
3788                **
3789                ** Name:    CRDFAB  -  Collapse File
3790                **
3791                ** Category:   LOCAL
3792                **
3793                ** Purpose:
```

```
3794                ;;      Destroy file created for card read & BUFDAL.
3795                ;;
3796                ;; Entry:
3797                ;;      R1 points at file data area (past header).
3798                ;;
3799                ;; Exit:
3800                ;;      Through MVMEM+.
3801                ;;
3802                ;; Calls:      BUFDAL, MVMEM+ (falls through).
3803                ;;
3804                ;; Uses.......
3805                ;;              A,B,C,D,P,D0,D1,R0,R1,R2,SCRTCH[4-0].
3806                ;;
3807                ;; Stk lvls:   3
3808                ;;
3809                ;; History:
3810                ;;
3811                ;;    Date      Programmer              Modification
3812                ;;  --------    ----------   ----------------------------------
3813                ;;  10/21/82    NM           Rewrote totally to the max
3814                ;;
3815                ;;*****************************************************************
3816                ;;*****************************************************************
3817 1D92B 8E2D  CRDFAB GOSUBL BUFDAL
           5F
3818 1D931 111          A=R1
3819 1D934 131          D1=A                    File pointer
3820 1D937 1C4          D1=D1- 5
3821 1D93A 1CF          D1=D1- 16
3822 1D93D 1CF          D1=D1- 16
3823 1D940 133          AD1EX
3824 1D943 D8           B=A    A                Start of this file
3825 1D945 1B17         D0=(5) =MAINEN
           5F2
3826 1D94C 142          A=DAT0 A
3827 1D94F CC           A=A-1  A
3828 1D951 CC           A=A-1  A                End of this file
3829 1D953 D6           C=A    A                Ditto
3830 1D955 E1           B=B-C  A                Size of move (negative)
3831 1D957 20    MVMEM+ P=     0
3832 1D959 8D00         GOVLNG =MVMEM+          Collapse this file
           000
3833                ;;*****************************************************************
3834                ;;*****************************************************************
3835                ;;
3836                ;; Name:    MAKHDR  -  Read In Card Header
3837                ;;
3838                ;; Category:   LOCAL
3839                ;;
3840                ;; Purpose:
3841                ;;      Copy card header field to RAM w/o compare.
3842                ;;
3843                ;; Entry:
3844                ;;      CARD IS MOVING!
3845                ;;      D0 points at spot to write card data to.
```

```
3846            **      R1 and R2 as used in CRDFIL.
3847            **
3848            ** Exit:
3849            **      Carry set if read failure (message given in WAIT
3850            **        routine).
3851            **
3852            ** Calls:       NOCOMP, RD8S, UPDCKP {unless error exit}.
3853            **
3854            ** Uses.......
3855            **              C[5-0],P,D1,B[B],S1
3856            **              {A,B,C,D,D0,R0,scratch RAM if error}
3857            **
3858            ** Stk lvls:    2
3859            **
3860            ** Detail:
3861            **      Used to read header from first card in CRDFIL and in
3862            **      CAT CARD.
3863            **
3864            ** Algorithm:
3865            **      Clear D for checksum.
3866            **      {in all reads below, RTNC if error}.
3867            **      Read first 4 bytes; write to RAM; update checksum.
3868            **      Read second 4 bytes.  If subformat (first byte of
3869            **        first read) # 0; goto 1.
3870            **      Write second read to RAM; update checksum.
3871            **      Read third 4 bytes.
3872            **      If byte 0 of third read {HP-75 filetype} <> "I" {LIF1}
3873            **        then error out with eBADFM.
3874            **      Goto 2.
3875            **    1: If subformat (first byte of first read, still) <> 1
3876            **        then error out with eBADFM.
3877            **      Write second read to RAM; update checksum.
3878            **      Read third 4 bytes.
3879            **    2: Write third read to RAM; update checksum.
3880            **      Read and write 20 more bytes; updating checksum.
3881            **      Compress checksum to 4 nibs with wraparound carry.
3882            **      Read final 4 bytes from card.  Add lower 2 bytes to
3883            **        checksum with wraparound carry.
3884            **      Compress checksum to 1 byte.
3885            **      Compare checksum to byte 2 of final read; RTNCC if
3886            **        matches.
3887            **      Error out with eRWERR.
3888            **
3889            ** History:
3890            **
3891            **    Date      Programmer          Modification
3892            **  --------    ----------  ----------------------------------
3893            **  06/25/82    NM          Added documentation
3894            **
3895            ********************************************************************
3896            ********************************************************************
3897 1D960 AF3   MAKHDR D=0     W               For checksum
3898 1D963 72B8          GOSUB   RD8S            First 4 bytes of header
3899 1D967 400           RTNC
3900 1D96A 79D1          GOSUB   NOCOMP          Write to ram
```

```
3901 1D96E 77A8          GOSUB  RD8S          Bytes 4-7 of header
3902 1D972 400           RTNC
3903 1D975 AEB           C=D    B
3904 1D978 A6E           C=C-1  B             HP-75 subformat card?
3905 1D97B 4D0           GOC    MAKH20        Yes.
3906 1D97E 96A           ?C=0   B             No.  HP-71 subformat?
3907 1D981 80            GOYES  MAKH20        Yes.
3908 1D983 8CAB          GOLONG UNKCD         No.  Unknown card.
           5F
3909 1D989 7AB1  MAKH20  GOSUB  NOCOMP        Write to ram
3910 1D98D 7888          GOSUB  RD8S          Bytes 8-11 of header
3911 1D991 400           RTNC
3912 1D994 7FA1          GOSUB  NOCOMP        Write bytes 8-11 to ram
3913 1D998 2F            P=     15
3914 1D99A 304           LCHEX  4
3915 1D99D AC5           B=C    S             Counter for 5 reads
3916 1D9A0 7578  MAKH90  GOSUB  RD8S          Read 4 bytes
3917 1D9A4 400           RTNC
3918 1D9A7 7C91          GOSUB  NOCOMP        Write to ram
3919 1D9AB A4D           B=B-1  S
3920 1D9AE 51F           GONC   MAKH90
3921 1D9B1 23            P=     3             No.  checksum is next.
3922 1D9B3 A9B           C=D    WP            Will compress to 4 nibs
3923 1D9B6 BF7           DSR    W
3924 1D9B9 BF7           DSR    W
3925 1D9BC BF7           DSR    W
3926 1D9BF F7            DSR    A
3927 1D9C1 7E81          GOSUB  UPDCKP        Add to checksum
3928 1D9C5 7058          GOSUB  RD8S          Final read in header
3929 1D9C9 400           RTNC
3930 1D9CC 1587          DAT0=A 8             Write it out
3931 1D9D0 23            P=     3
3932 1D9D2 A9B           C=D    WP
3933 1D9D5 A12           C=A+C  WP            Add 2 more bytes to cksum
3934 1D9D8 550           GONC   MAK120
3935 1D9DB B16           C=C+1  WP
3936 1D9DE AE7   MAK120  D=C    B
3937 1D9E1 F6            CSR    A
3938 1D9E3 BB6           CSR    X
3939 1D9E6 A6B           C=C+D  B             Compress to 2 nibs
3940 1D9E9 BF4           ASR    W
3941 1D9EC BF4           ASR    W
3942 1D9EF BF4           ASR    W
3943 1D9F2 F4            ASR    A
3944 1D9F4 962           ?A=C   B             Header checksum matches?
3945 1D9F7 80            GOYES  MAK125        Yes
3946 1D9F9 8C45          GOLONG RWERR
           5F
3947 1D9FF 03    MAK125  RTNCC
3948               ********************************************************************
3949               ********************************************************************
3950               **
3951               ** Name:    OFFSET  -  Compute Offset Of Current Track
3952               **
3953               ** Category:   LOCAL
```

```
3954            **
3955            ** Purpose:
3956            **      Compute offset of current card (relative to start of
3957            **      data area).
3958            **
3959            ** Entry:
3960            **      R2 points at card header I/Obuffer.
3961            **
3962            ** Exit:
3963            **      Offset in A[A].
3964            **      Size of this track in C[A].
3965            **      D0 points at tracksize in card header I/Obuffer.
3966            **      P=0.
3967            **      Carry clear.
3968            **
3969            ** Calls:      LC2TRK.
3970            **
3971            ** Uses.......
3972            **             A[A],B[A],C,P,D0.
3973            **
3974            ** Stk lvls:   1
3975            **
3976            ** Detail:
3977            **      Uses a fast multiply algorithm.  This code is called
3978            **      from CRDFIL in a time-critical section of code.
3979            **
3980            ** Algorithm:
3981            **      A[B]=trk#-1.
3982            **      A[XS]=1 if A[B] is odd, else A[XS]=0.
3983            **      Set low bit in A[B] {this adds 1 if A[B] is even}.
3984            **      C[A]=trksize in nibbles.
3985            **      B[A]=0 {for result}.
3986            **   1: Double result {B}.
3987            **      Double A[B].  If carry, B=B+C.
3988            **      If A<>0 goto 1.
3989            **      If A[XS]=1 {we did one too many adds}, subtract C from
3990            **        result.  Put result in A.
3991            **      Point D0 at trksize field in card header.
3992            **      Read trksize field from card header into C.
3993            **      RTN.
3994            **
3995            ** History:
3996            **
3997            **    Date     Programmer             Modification
3998            **  --------   ----------   ----------------------------------
3999            **  06/25/82   NM           Added documentation
4000            **
4001            ******************************************************************
4002            ******************************************************************
4003 1DA01 11A   OFFSET C=R2
4004 1DA04 134          D0=C                    Point at header
4005 1DA07 161          D0=D0+ 2
4006 1DA0A 14A          A=DAT0 B                Read track#
4007 1DA0D A6C          A=A-1  B                Will mpy trk#-1 by trksize
4008 1DA10 20           P=     0
```

```
4009 1DA12 301          LCHEX  1
4010 1DA15 0E02         C=A&C  P
4011 1DA19 BE2          CSL    B
4012 1DA1C BB2          CSL    X
4013 1DA1F AAA          A=C    XS              Odd(a[b])
4014 1DA22 301          LCHEX  1
4015 1DA25 0E0E         A=A!C  P               Set low bit in a[b] for marker
4016 1DA29 7E61         GOSUB  LC2TRK          Trksize in nibs
4017 1DA2D D1           B=0    A               Room for result
4018 1DA2F C5    MPTK10 B=B+B  A               Shift result
4019 1DA31 A64          A=A+A  B               Shift multiplier
4020 1DA34 540          GONC   MPTK20          =1?
4021 1DA37 C1           B=B+C  A               Yes, add trksize
4022 1DA39 96C   MPTK20 ?A#0   B               Anything left of multiplier?
4023 1DA3C 3F           GOYES  MPTK10          Yes, continue loop
4024 1DA3E DC           ABEX   A
4025 1DA40 92D          ?B#0   XS              Was a[b] odd?
4026 1DA43 40           GOYES  MPTK30          Yes, a[a] is result
4027 1DA45 EA           A=A-C  A               No.  did 1 too many adds.
4028 1DA47 163   MPTK30 D0=D0+ 2*2             Point at tracksize
4029 1DA4A D2    DONIBC C=0    A
4030 1DA4C 15E3         C=DAT0 4               Size of this track
4031 1DA50 C6           C=C+C  A               In nibs
4032 1DA52 01           RTN
4033          ************************************************************************
4034          ************************************************************************
4035          **
4036          ** Name:    SETBIT  -  Set A Bit In The Bitmap
4037          **
4038          ** Category:   LOCAL
4039          **
4040          ** Purpose:
4041          **      Set a bit to mark a track as having been read.
4042          **
4043          ** Entry:
4044          **      R1[4-0]=address of file data area,
4045          **        [9-5]=size of data area.
4046          **      C[B] = Bit#.
4047          **
4048          ** Exit:
4049          **
4050          ** Calls:      LOCBIT.
4051          **
4052          ** Uses.......
4053          **             A,C,P,D0.
4054          **
4055          ** Stk lvls:   2
4056          **
4057          ** Detail:
4058          **      Bitmap is at the end of the file data area.
4059          **
4060          ** Algorithm:
4061          **      LOCBIT.
4062          **      AND nibble with bitmask.
4063          **      Write out nibble.
```

```
4064            **
4065            ** History:
4066            **
4067            **    Date     Programmer            Modification
4068            **  --------   ----------   ----------------------------------
4069            **  06/25/82   NM           Added documentation
4070            **
4071            ********************************************************************
4072            ********************************************************************
4073 1DA54 7601  SETBIT GOSUB  LOCBIT
4074 1DA58 0E4E         A=A!C  S              Or bitmask with nibble
4075 1DA5C 1504         DAT0=A S
4076 1DA60 01           RTN
4077            ********************************************************************
4078            ********************************************************************
4079            **
4080            ** Name:    HDRHDR  -  Decode Encoded Filetype
4081            **
4082            ** Category:   LOCAL
4083            **
4084            ** Purpose:
4085            **      Convert filetype from card header into unencoded
4086            **      filetype and security/privacy bits.
4087            **
4088            ** Entry:
4089            **      D0 points at beginning of card header I/Obuffer (i.e,
4090            **        at SUBFORMAT byte).
4091            **
4092            ** Exit:
4093            **      If carry set,
4094            **          Filetype unrecognized (not found in FTYPF#) and
4095            **            not in standard range,
4096            **          A[A]=filetype,
4097            **          C[B]=flags (10 in this case).
4098            **        else
4099            **          Filetype recognized (found by FTYPF#) or, if not
4100            **            recognized, is in standard range,
4101            **          Unencoded filetype in A[A],
4102            **          Flags in C[B],
4103            **          If XM=0 then
4104            **              Filetype recognized or, if not, in standard
4105            **                range and not private.
4106            **            else
4107            **              Filetype in standard range and private.
4108            **
4109            ** Calls:      FTYPF#, STDRG?.
4110            **
4111            ** Uses.......
4112            **              A,B,C,D0,R0,P,XM.
4113            **
4114            ** Stk lvls:   3
4115            **
4116            ** Detail:
4117            **      Filetype is read from header pointed to by D0.
4118            **      If filetype if found by FTYPF#, the security nibble is
```

```
4119                **        taken as (position of filetype relative to unencoded
4120                **        filetype in table)-1.  This quantity is returned by
4121                **        FTYPF#.
4122                **
4123                ** Algorithm:
4124                **      If subformat {in card header}=0 then A[A]=00001, else
4125                **        A[A]=filetype from card header.
4126                **      Search for filetype in filetype table {FTYPF#}.
4127                **      Clear XM.
4128                **      If found goto 1.
4129                **      If file is not in standard range goto 2.
4130                **      A[A]=filetype - filetype mod 4.
4131                **      C[B]=filetype mod 4.
4132                **      If privacy bit set then set XM.
4133                **      RTNCC.
4134                **    1: Read copy code from ftype table {located at (C[A]+1)}.
4135                **      Security nibble = position of ftype in ftype group-1.
4136                **      Put above 2 nibbles in C[B].
4137                **      Read primary {unencoded} filetype from ftype table into
4138                **        A[3-0].
4139                **      RTNCC.
4140                **    2: A[A]=filetype passed to this routine.
4141                **      C[B]=10 {copy code 1, security 0}.
4142                **      RTNSC {indicate unrecognized}.
4143                **
4144                ** History:
4145                **
4146                **    Date      Programmer            Modification
4147                **  --------    ----------   ------------------------------------
4148                **  06/25/82    NM           Added documentation
4149                **
4150                ************************************************************
4151                ************************************************************
4152 1DA62 1524  HDRHDR A=DAT0 S              Read subformat lonib
4153 1DA66 16D          DO=DO+ 2*7
4154 1DA69 D0           A=0    A
4155 1DA6B 15A3         A=DAT0 4              Read filetype
4156 1DA6F 94C          ?A#0   S              Subformat 0?
4157 1DA72 81           GOYES  HDRH10         No
4158 1DA74 20           P=     0
4159 1DA76 3194         LCASC  \I\
4160 1DA7A F4           ASR    A
4161 1DA7C F4           ASR    A
4162 1DA7E B62          C=C-A  B              C=HP-75 filetype - \I\
4163 1DA81 D0           A=0    A
4164 1DA83 96E          ?C#0   B              Is this LIF1?
4165 1DA86 40           GOYES  HDRH10         No.  Filetype=0000.
4166 1DA88 E4           A=A+1  A              Yes.  type=0001
4167 1DA8A 8F00  HDRH10 GOSBVL =FTYPF#        Look for filetype in table
          000
4168 1DA91 821          XM=0                  Clear XM for result
4169 1DA94 463          GOC    HDRH50         Go if filetype found
4170 1DA97 8ECB         GOSUBL STDRG?         File in standard range?
          3F
4171 1DA9D 4B4          GOC    HDRH60         No
```

```
4172 1DAA0 30C          LCHEX  C
4173 1DAA3 A88          B=A    P              Copy lonib of filetype
4174 1DAA6 0E06         A=A&C  P              Unencoded filetype
4175 1DAAA 303          LCHEX  3
4176 1DAAD 0E01         B=B&C  P              Security & privacy bits
4177 1DAB1 302          LCHEX  2
4178 1DAB4 0E05         C=B&C  P              Privacy bit
4179 1DAB8 90A          ?C=0   P              Privacy set?
4180 1DABB 60           GOYES  HDRH40         No
4181 1DABD 7620         GOSUB  SXM            Set xm
4182 1DAC1 A89   HDRH40 C=B    P              Copy security bit
4183 1DAC4 21           P=     1
4184 1DAC6 301          LCHEX  1              Copy code=1
4185 1DAC9 03           RTNCC
4186 1DACB 134   HDRH50 DO=C                  Point at table entry
4187 1DACE 160          DO=DO+ 1
4188 1DAD1 15E0         C=DAT0 1              Read copy code
4189 1DAD5 AC9          C=B    S              Position in table
4190 1DAD8 A4E          C=C-1  S              Security
4191 1DADB 812          CSLC                  Flags in c[b]
4192 1DADE 16E          DO=DO+ 15             Point at primary filetype#
4193 1DAE1 15A3         A=DAT0 4
4194 1DAE5 03           RTNCC
4195 1DAE7 00    SXM    RTNSXM
4196 1DAE9 3101  HDRH60 LCHEX  10             Dummy flags
4197 1DAED 02           RTNSC                 Indicate unrecognized
4198            ********************************************************************
4199            ********************************************************************
4200            **
4201            ** Name:(S) FNDCLR  -  Find Next Clear Bit In Bitmap
4202            **
4203            ** Category:   LOCAL
4204            **
4205            ** Purpose:
4206            **      Find trk# of next unread side.
4207            **
4208            ** Entry:
4209            **      Current bit# in C[B].
4210            **      R1[4-0]=address of data area.
4211            **        [9-5]=size of data area.
4212            **
4213            ** Exit:
4214            **      Number of next clear bit in C[B].
4215            **      Carry set if no clear bits (i.e., bit counter incre-
4216            **        ments past 255 in its search for clear bit).
4217            **
4218            ** Calls:      LOCBIT.
4219            **
4220            ** Uses.......
4221            **             A,B,C,P,DO.
4222            **
4223            ** Stk lvls:   2
4224            **
4225            ** Detail:
4226            **      Begins search for clear bit from CURRENT bit# (i.e.,
```

```
4227            **      find NEXT unread track after this one).
4228            **
4229            ** Algorithm:
4230            **      Locate bit# passed in C.
4231            **   1: Complement nibble containing bit# (A[S]).
4232            **      Hold nibble in B[S].
4233            **   2: And bitmask with nibble.  If result <> 0 return bit#
4234            **        in C.
4235            **      Increment bit#.
4236            **      Restore nibble to A[S].
4237            **      Move bitmask bit left; goto 2 if not carry out.
4238            **      Increment D0; Read next nibble; mask=1; goto 1.
4239            **
4240            ** History:
4241            **
4242            **    Date      Programmer            Modification
4243            **  --------   ----------   ----------------------------------
4244            **  06/25/82   NM           Added documentation
4245            **
4246            ********************************************************************
4247            ********************************************************************
4248 1DAEF AE5  =FNDCLR B=C    B               Hold bit#
4249 1DAF2 7860         GOSUB  LOCBIT          Find nibble
4250 1DAF6 BCC   FIND05 A=-A-1 S               Complement nibble.
4251 1DAF9 AC8   FIND10 B=A    S               Hold nibble
4252 1DAFC 0E46  FIND20 A=A&C  S               And nibble with mask
4253 1DB00 94C          ?A#0   S               Found set bit?
4254 1DB03 E1           GOYES  FIND30          Yes
4255 1DB05 B65          B=B+1  B               No, increment bit#
4256 1DB08 4D1          GOC    FIND40          Return if fail
4257 1DB0B AC4          A=B    S               Restore nibble
4258 1DB0E A46          C=C+C  S               Move bitmask
4259 1DB11 5AE          GONC   FIND20          Go if not carry out
4260 1DB14 160          D0=D0+ 1               Time to examine next nibble
4261 1DB17 1524         A=DAT0 S
4262 1DB1B B46          C=C+1  S               New bitmask
4263 1DB1E 57D          GONC   FIND05          B.E.T.
4264 1DB21 AE9   FIND30 C=B    B
4265 1DB24 03    FIND35 RTNCC
4266 1DB26 AE9   FIND40 C=B    B
4267 1DB29 02           RTNSC
4268            ********************************************************************
4269            ********************************************************************
4270            **
4271            ** Name:    CMPALL  -  Copy With Compare
4272            ** Name:    CMPWRT  -  Copy With Compare
4273            ** Name:    NOCOMP  -  Copy Without Compare
4274            **
4275            ** Category:   LOCAL
4276            **
4277            ** Purpose:
4278            **      Write information to RAM while comparing with current
4279            **      contents.  Specifically:
4280            **        CMPALL: Compare entire 8-nibble read before writing.
4281            **        CMPWRT: Compare specified bits before writing.
```

```
4282                **        NOCOMP: Write to RAM w/o compare.
4283                **
4284                ** Entry:
4285                **   (all):  A[7-0] = data to write to RAM.
4286                **           Running checksum in D[7-0].
4287                **           D0 points at spot to write.
4288                **   CMPWRT: C[7-0] = bitmap of bits to compare.
4289                **
4290                ** Exit:
4291                **      Carry set if compare failure (write was not performed).
4292                **      Else data was written to RAM,
4293                **        D0=D0+8,
4294                **        Checksum in D was updated,
4295                **        A=C=Data originally passed in A.
4296                **        P=7.
4297                **
4298                ** Calls:      None.
4299                **
4300                ** Uses.......
4301                **            B[7-0],C,D[7-0],P,D0.
4302                **
4303                ** Stk lvls:   0
4304                **
4305                ** Detail:
4306                **      Used when card header is being read in.  Certain fields
4307                **      within the header must be the same from card to card,
4308                **      and this provides the mechanism for checking that.
4309                **      If the compare fails, the write is NOT performed.
4310                **      That way, the original information is not obliterated.
4311                **      This code also maintains the checksum which will be
4312                **      used after the entire header is read in.
4313                **
4314                ** Algorithm:
4315                **   CMPALL: C[7-0]=FFFFFFFF
4316                **   CMPWRT: B=C {comparison bitmask}.
4317                **           C=DAT0 {read data to be compared to}.
4318                **           And bitmask into C; And A into bitmask.
4319                **           If B<>C then return carry set {error}.
4320                **   NOCOMP: Write A[7-0] at D0.
4321                **           Increment D0 by 8.
4322                **           Update checksum in D[7-0].
4323                **           RTN.
4324                **
4325                ** History:
4326                **
4327                **    Date      Programmer          Modification
4328                **  --------   ----------   ----------------------------------
4329                **  06/25/82   NM           Added documentation
4330                **
4331                ****************************************************************
4332                ****************************************************************
4333 1DB2B AF2   CMPALL C=0     W              CMPALL=CMPWRT with C=FFFFFFFF
4334 1DB2E A7E         C=C-1   W
4335 1DB31 27    CMPWRT P=      7
4336 1DB33 A95         B=C     WP             Hold compare mask
```

```
4337 1DB36 15E7          C=DATO 8              Data to be compared
4338 1DB3A 0E15          C=B&C  WP             And with compare mask
4339 1DB3E 0E14          B=A&B  WP             And new data with compare mask
4340 1DB42 915           ?B#C   WP             Unmasked bits compare true?
4341 1DB45 00            RTNYES                No, compare error
4342 1DB47 1587  NOCOMP  DATO=A 8              Write new data
4343 1DB4B 167           D0=D0+ 8
4344 1DB4E 27            P=     7
4345 1DB50 A96           C=A    WP
4346 1DB53 A13   UPDCKP  D=D+C  WP             Update checksum
4347 1DB56 500           RTNNC
4348 1DB59 B17           D=D+1  WP             Wraparound carry
4349 1DB5C 01            RTN
4350             ********************************************************************
4351             ********************************************************************
4352             **
4353             ** Name:    LOCBIT  -  Locate Bit In The Bitmap
4354             **
4355             ** Category:   LOCAL
4356             **
4357             ** Purpose:
4358             **       Locate desired bit in the tracks-read bitmap.
4359             **
4360             ** Entry:
4361             **       Bit# in C[B] (1-256).
4362             **       R1[4-0]=address of data area.
4363             **         [9-5]=size of data area.
4364             **
4365             ** Exit:
4366             **       If carry set, routine was passed C[B]=0.
4367             **       Else A[S] contains the nibble containing the requested
4368             **         bit; C[S] contains a bitmask for the requested bit
4369             **         within A[S]; D0 points at the location where the
4370             **         nibble lives.
4371             **
4372             ** Calls:      CSRW5.
4373             **
4374             ** Uses.......
4375             **             A[A],A[S],C,P,D0.
4376             **
4377             ** Stk lvls:   1
4378             **
4379             ** Detail:
4380             **       Location of bitmap is (R1[4-0] + R1[9-5]).
4381             **
4382             ** Algorithm:
4383             **       D0=location of bitmap + (bit# div 4).
4384             **       Bitmask=2^(bit# mod 4).
4385             **       Read nibble at D0 into A[S].
4386             **       Put bitmask in C[S].
4387             **
4388             ** History:
4389             **
4390             **    Date     Programmer              Modification
4391             **  --------   ----------   ----------------------------------
```

```
4392                ** 06/25/82   NM          Added documentation
4393                **
4394                ********************************************************************
4395                ********************************************************************
4396 1DB5E D0     LOCBIT A=0    A
4397 1DB60 A6E           C=C-1  B              Range=0-255
4398 1DB63 400           RTNC
4399 1DB66 AEA           A=C    B
4400 1DB69 20            P=     0
4401 1DB6B 303           LCHEX  3
4402 1DB6E 0E02          C=A&C  P              Isolate lower 2 bits for mask
4403 1DB72 C4            A=A+A  A
4404 1DB74 C4            A=A+A  A
4405 1DB76 F4            ASR    A              Nibble#
4406 1DB78 B8E           C=-C-1 P              Complement bit# for magic lc
4407 1DB7B 80D0          P=C    0
4408 1DB7F 119           C=R1                  Address of sod
4409 1DB82 CA            A=A+C  A              Offset from sod to nibble
4410 1DB84 8EA7          GOSUBL csrw5          Size of data field
          6F
4411 1DB8A C2            C=C+A  A
4412 1DB8C 134           DO=C                  Point to desired nibble
4413 1DB8F 1524          A=DATO S              Read addressed nibble
4414 1DB93 3312          LCHEX  8421           Magic lc for bitmask
          48
4415 1DB99 01            RTN
4416                ********************************************************************
4417                ********************************************************************
4418                **
4419                ** Name:    LC2TRK  -  Load A Constant
4420                **
4421                ** Category:   LOCAL
4422                **
4423                ** Purpose:
4424                **      Load constant corresponding to trksize in nibbles.
4425                **
4426                ** Entry:
4427                **
4428                ** Exit:
4429                **      P=0.
4430                **      C[A]=Tracksize in nibbles.
4431                **
4432                ** Calls:      None.
4433                **
4434                ** Uses.......
4435                **             P,C[A].
4436                **
4437                ** Stk lvls:   0
4438                **
4439                ** History:
4440                **
4441                **    Date     Programmer             Modification
4442                **  --------   ----------   ------------------------------------
4443                **  10/07/82   NM           Proudly authored.
4444                **
```

```
4445            ****************************************************************
4446            ****************************************************************
4447 1DB9B 20    LC2TRK P=     0
4448 1DB9D 3441          LC(5)  2*(TRKSIZ)
           500
4449 1DBA4 01            RTN
4450            ****************************************************************
4451            ****************************************************************
4452            **
4453            ** Name:(S) pWCRD   -  Card Write Poll
4454            **
4455            ** Category:   POLL
4456            **
4457            ** Type:       FPOLL
4458            **
4459            ** Purpose:
4460            **      Allow processing before writing out a card track.
4461            **
4462            ** Should poll be "Handled" (return with XM=0)?:
4463            **      If polling should terminate, then poll should be
4464            **      handled.
4465            **
4466            ** Meaning of "Handling" Poll (what does code do if handled?):
4467            **      Code does nothing different if poll is handled.
4468            **      Handling merely terminates polling, which is probably
4469            **      the desired result.
4470            **
4471            ** Entry conditions for handler (registers, ST, RAM, etc.):
4472            **      We are about to prompt for a card.
4473            **      Carry set on entry.
4474            **      B[A] = Poll number.
4475            **      HEX mode.
4476            **      P=0.
4477            **      R1-R2 set up as FILCRD documentation explains.
4478            **      The bCARD buffer contains the card header.
4479            **
4480            ** Normal exit conditions from handler if handled (ST, RAM,
4481            ** registers, etc.):
4482            **      HEX mode.
4483            **      XM=0.
4484            **      Card header modified as desired.
4485            **
4486            ** Normal exit conditions from handler if not handled (ST, RAM,
4487            ** registers, etc.):
4488            **      HEX mode.
4489            **      XM=1.
4490            **      Card header modified if desired.
4491            **
4492            ** Available subroutine levels:
4493            **      2
4494            **
4495            ** NOTE:
4496            **      If you modify the card header, you must recompute the
4497            **      card header checksum, or you will never be able to
4498            **      read back the card you have written.
```

```
4499            **
4500            ** What registers/RAM may be used if handled?:
4501            **      A-D, D0, D1, P, R0, R3, R4, all scratch RAM.
4502            **
4503            ** What registers/RAM may be used if not handled?:
4504            **      A-C, D[5-15], D0, D1, P, R0, R3, R4, all scratch RAM.
4505            **
4506            ** Envisioned application(s):
4507            **      Setting up card header for partial card recovery.
4508            **      It is highly doubtful whether partial card recovery
4509            **      can be done, but this is the hook which allows you to
4510            **      try it.  The documentation for FNDPRT explains the
4511            **      meaning of the partial card recovery information
4512            **      fields.  Good luck.
4513            **
4514            ** History:
4515            **
4516            **    Date     Programmer             Modification
4517            **  --------   ----------   ----------------------------------
4518            **  08/01/83   NM           Added documentation
4519            **
4520            ********************************************************************
4521            ********************************************************************
4522            ********************************************************************
4523            ********************************************************************
4524            **
4525            ** Name:(S) pRCRD   -  Poll After Reading Card.
4526            **
4527            ** Category:   POLL
4528            **
4529            ** Type:       FPOLL
4530            **
4531            ** Purpose:
4532            **      Poll after each card track is read.
4533            **
4534            ** Should poll be "Handled" (return with XM=0)?:
4535            **      If it is desired to terminate polling, yes.
4536            **
4537            ** Meaning of "Handling" Poll (what does code do if handled?):
4538            **      Code doesn't do anything different if poll is handled.
4539            **      Handling simply stops polling, which may be desirable.
4540            **
4541            ** Entry conditions for handler (registers, ST, RAM, etc.):
4542            **      Carry set on entry.
4543            **      B[A] = Poll number.
4544            **      HEX mode.
4545            **      P=0.
4546            **      R1, R2 as defined in CRDFIL header.
4547            **      bCARD buffer contains header of card just read in.
4548            **      Code has just read a track and is about put up a
4549            **        "trk <nnn> done" message.
4550            **
4551            ** Normal exit conditions from handler if handled (ST, RAM,
4552            ** registers, etc.):
4553            **      HEX mode.
```

```
4554            **      XM=0.
4555            **
4556            ** Normal exit conditions from handler if not handled (ST, RAM,
4557            ** registers, etc.):
4558            **      HEX mode.
4559            **      XM=1.
4560            **
4561            ** Available subroutine levels:
4562            **      3
4563            **
4564            ** What registers/RAM may be used if handled?:
4565            **      A-D, D0, D1, P, R0, R3, R4.
4566            **      All scratch RAM.
4567            **
4568            ** What registers/RAM may be used if not handled?:
4569            **      A-C, C[5-15], D0, D1, P, R0, R3, R4.
4570            **      All scratch RAM.
4571            **
4572            ** Special memory/pointer considerations (are pointers funny?):
4573            **      There is no available memory.
4574            **
4575            ** Envisioned application(s):
4576            **      This is supposed to be the hook to allow partial card
4577            **      recovery.  I am skeptical, but I'll keep it to myself.
4578            **      If the card was written by somebody who knows how to
4579            **      do partial card recovery, the header will contain data
4580            **      necessary to perform recovery.  This poll is an
4581            **      opportunity to take the data and stuff it somewhere
4582            **      useful.  One recovery scheme which worked very well
4583            **      is the past was storing the data in the space to be
4584            **      occupied by adjacent tracks IF the adjacent track has
4585            **      not been read yet.  The flaw in this is what happens if
4586            **      that data is munched by an unsuccessful read in the
4587            **      adjacent track.  The data is lost.  So what to do?
4588            **      Maybe create an I/O buffer to hold the data.  Of course
4589            **      that buffer had better be around before the read is
4590            **      initiated, since the read code sucks up all available
4591            **      memory to make room for the biggest card set possible.
4592            **      Good luck.
4593            **
4594            ** History:
4595            **
4596            **    Date     Programmer             Modification
4597            **  --------   ----------   ----------------------------------
4598            **  08/01/83   NM           Added documentation
4599            **
4600            ********************************************************************
4601            ********************************************************************
4602            ********************************************************************
4603            ********************************************************************
4604            **
4605            ** Name:(S) pCRDAB  -  ABORT Card Read Poll
4606            **
4607            ** Category:   POLL
4608            **
```

```
4609            ** Type:       FPOLL
4610            **
4611            ** Purpose:
4612            **      Poll upon ATTN-key or timeout abort of card read
4613            **      operation.
4614            **
4615            ** Should poll be "Handled" (return with XM=0)?:
4616            **      Yes, if...
4617            **
4618            ** Meaning of "Handling" Poll (what does code do if handled?):
4619            **      ... handler has cleanly terminated card read operation.
4620            **      This means collapsing the file to the proper size
4621            **      (which may be zero).  If poll is handled, card reader
4622            **      code does not collapse file.
4623            **
4624            ** Entry conditions for handler (registers, ST, RAM, etc.):
4625            **      Carry set on entry.
4626            **      B[A] = Poll number.
4627            **      HEX mode.
4628            **      P=0.
4629            **      R1 and R2 have meaning as explained in CRDFIL header.
4630            **
4631            ** Normal exit conditions from handler if handled (ST, RAM,
4632            ** registers, etc.):
4633            **      HEX mode.
4634            **      XM=0.
4635            **
4636            ** Normal exit conditions from handler if not handled (ST, RAM,
4637            ** registers, etc.):
4638            **      HEX mode.
4639            **      XM=1.
4640            **
4641            ** Available subroutine levels:
4642            **      3
4643            **
4644            ** What registers/RAM may be used if handled?:
4645            **      A-D, D0, D1, P, R0-R4, all scratch RAM.
4646            **
4647            ** What registers/RAM may be used if not handled?:
4648            **      A-C, D[15-5] D0, D1, P, R0, R3, R4, all scratch RAM.
4649            **
4650            ** Special memory/pointer considerations (are pointers funny?):
4651            **      There is no available memory.
4652            **
4653            ** Envisioned application(s):
4654            **      This is a chance to do partial card recovery with all
4655            **      that neat information saved during the pRCRD poll.
4656            **      See that documentation for appropriate caveats.
4657            **
4658            ** History:
4659            **
4660            **    Date     Programmer          Modification
4661            **  --------   ----------   ----------------------------------
4662            **  08/01/83   NM           Added documentation
4663            **
```

```
4664            ********************************************************************
4665            ********************************************************************
4666 1DBA6              END
```

```
 ACBAT?  Ext                  -  1925
 ALIG10  Abs  118868 #1D054   -  1988  2000
 ALIG25  Abs  118913 #1D081   -  2001  1994
 ALIG35  Abs  118920 #1D088   -  2003  1996  1998
 ALIGN   Abs  118864 #1D050   -  1987   478   525  3541  3652
 ASLW5   Ext                  -  2872
 ASRW4   Ext                  -  3361
 ASRW5   Ext                  -  2873
 BADHDR  Abs  120127 #1D53F   -  3313  3322  3329  3338
 BLANKC  Ext                  -   403
 BUFDAL  Abs  118531 #1CF03   -  1640   356   431  1528  1949  3737  3817
 CAT$20  Ext                  -  3720
 CAT$66  Ext                  -  3739
 CAT$C1  Abs  120932 #1D864   -  3684  3665
 CAT$C2  Abs  121091 #1D903   -  3737  3727
 CAT100  Ext                  -  3729
 CATC05  Abs  120785 #1D7D1   -  3643  3639
 CATC10  Abs  120806 #1D7E6   -  3649  3647
 CATC20  Abs  120812 #1D7EC   -  3651  3657  3662
 CATC30  Abs  120827 #1D7FB   -  3655  3653
 CATC40  Abs  120887 #1D837   -  3672  3680
 CATC50  Abs  120918 #1D856   -  3681  3674
=CATCRD  Abs  120776 #1D7C8   -  3641
 CERR    Abs  119221 #1D1B5   -  2812  2798
 CHKS10  Abs  118263 #1CDF7   -  1309  1314  1316
 CHKS20  Abs  118287 #1CE0F   -  1317  1310
 CHKS30  Abs  118308 #1CE24   -  1324  1318  1322
 CHKS40  Abs  118313 #1CE29   -  1326  1328
 CHKS70  Abs  118343 #1CE47   -  1337  1330
 CHKSUM  Abs  118254 #1CDEE   -  1306   411   458   469  3173
 CLRALL  Abs  121107 #1D913   -  3779  3101
 CMPALL  Abs  121643 #1DB2B   -  4333  3328
 CMPTIM  Ext                  -   302
 CMPWRT  Abs  121649 #1DB31   -  4335  3311  3321  3337
 CR      Abs  180224 #2C000   -    15  1484  2071  2073  2777  2789  2822
 CR??    Abs  118420 #1CE94   -  1476   261  3049  3535  3641
 CR??10  Abs  118429 #1CE9D   -  1479  3640
 CR??20  Abs  118437 #1CEA5   -  1481  1478
 CRD130  Abs  119570 #1D312   -  3136  3128  3130  3372
 CRD140  Abs  119613 #1D33D   -  3151  3149  3157
 CRD150  Abs  119636 #1D354   -  3158  3152
 CRD155  Abs  119652 #1D364   -  3163  3154
 CRD160  Abs  119666 #1D372   -  3167  3161
 CRD180  Abs  119721 #1D3A9   -  3185  3181
 CRD220  Abs  119994 #1D4BA   -  3270  3268
 CRD222  Abs  120015 #1D4CF   -  3278  3274
 CRD225  Abs  120047 #1D4EF   -  3289  3281
 CRD230  Abs  120066 #1D502   -  3295  3293
 CRD240  Abs  120070 #1D506   -  3296  3288
 CRD270  Abs  120139 #1D54B   -  3316  3312
 CRD280  Abs  120170 #1D56A   -  3326  3331
 CRD285  Abs  120194 #1D582   -  3333  3340  3343
 CRD290  Abs  120248 #1D5B8   -  3350  3348
 CRD300  Abs  120268 #1D5CC   -  3357  3355
 CRD310  Abs  120313 #1D5F9   -  3372  3363
```

```
 CRD320   Abs  120389 #1D645 -  3400  3420
 CRD390   Abs  120444 #1D67C -  3421
 CRD400   Abs  120446 #1D67E -  3422  3391  3411
 CRD410   Abs  120494 #1D6AE -  3439  3434
 CRD420   Abs  120538 #1D6DA -  3456  3440
 CRDDON   Abs  120317 #1D5FD -  3373  3289
 CRDER?   Abs  120135 #1D547 -  3315  3302  3308  3317  3327  3333
 CRDERR   Abs  119717 #1D3A5 -  3183  3163
 CRDF13   Abs  119427 #1D283 -  3089  3086
 CRDF20   Abs  119484 #1D2BC -  3108  3117  3122  3133
 CRDF30   Abs  119499 #1D2CB -  3113  3109
 CRDFAB   Abs  121131 #1D92B -  3817  3087  3110  3210  3224  3228  3246
=CRDFIL   Abs  119325 #1D21D -  3049
 CRDFNM   Abs  119419 #1D27B -  3087  3100  3150  3248
 CRDN10   Abs  119774 #1D3DE -  3202  3196
 CRDN20   Abs  119813 #1D405 -  3213  3204  3209
 CRDNXT   Abs  119743 #1D3BF -  3193  3184  3315  3365
 CRDOFF   Abs  118963 #1D0B3 -  2115   523   565  1863  3167  3549  3663
 CREATF   Ext                -  3062
 CRHD10   Abs  119854 #1D42E -  3226  3223
 CRHD35   Abs  120051 #1D4F3 -  3290  3297
 CRHD60   Abs  119871 #1D43F -  3231  3227
 CRHD70   Abs  119932 #1D47C -  3249  3219  3241  3245
 CRLFND   Ext                -  1531
 CRPRS?   Abs  118441 #1CEA9 -  1484  1477  3638
 CSLW5    Ext                -   392  1773
 CSRW5    Ext                -  2871
=CT$CRD   Abs  120758 #1D7B6 -  3637
 DONIBC   Abs  121418 #1DA4A -  4029  3252  3699
 D1+13B   Abs  118501 #1CEE5 -  1566   295  3124  3199
 D1+21B   Abs  118498 #1CEE2 -  1565   399
 D1+27B   Abs  118495 #1CEDF -  1564  1093
 D1+29B   Abs  118492 #1CEDC -  1563   459  3176
 DAYYMD   Ext                -   307
 DSPCHA   Ext                -  3679
 DSPDLY   Ext                -  3730
 DTERR    Abs  119206 #1D1A6 -  2805  2821
 FIL100   Abs  117572 #1CB44 -   489   493   496   498   501   503   505   508
 FIL120   Abs  117576 #1CB48 -   490   486
 FIL130   Abs  117639 #1CB87 -   508   519
 FIL140   Abs  117656 #1CB98 -   514   512
 FIL150   Abs  117667 #1CBA3 -   518   521
 FILC05   Abs  116878 #1C88E -   266   264
 FILC07   Abs  117092 #1C964 -   331   326
 FILC10   Abs  117139 #1C993 -   347   345
 FILC12   Abs  117174 #1C9B6 -   357   348
 FILC14   Abs  117214 #1C9DE -   372   369
 FILC15   Abs  117249 #1CA01 -   384   379
 FILC16   Abs  117315 #1CA43 -   405   401
 FILC18   Abs  117319 #1CA47 -   406   404
 FILC40   Abs  117357 #1CA6D -   420   579
 FILC45   Abs  117393 #1CA91 -   432   428
 FILC50   Abs  117425 #1CAB1 -   442   436   440
 FILC60   Abs  117448 #1CAC8 -   450   448
 FILC70   Abs  117458 #1CAD2 -   454   451
```

```
 FILC75  Abs  117531 #1CB1B -   477   483   489   539
 FILC76  Abs  117546 #1CB2A -   481   479
 FILC90  Abs  117557 #1CB35 -   484
 FILCAB  Abs  117170 #1C9B2 -   356   354
=FILCRD  Abs  116857 #1C879 -   259
 FILEF   Ext                 -  3203
 FINCRD  Abs  117381 #1CA85 -   429  3470
 FIND05  Abs  121590 #1DAF6 -  4250  4263
 FIND10  Abs  121593 #1DAF9 -  4251
 FIND20  Abs  121596 #1DAFC -  4252  4259
 FIND30  Abs  121633 #1DB21 -  4264  4254
 FIND35  Abs  121636 #1DB24 -  4265
 FIND40  Abs  121638 #1DB26 -  4266  4256
 FINDAj  Ext                 -  1992
=FNDCLR  Abs  121583 #1DAEF -  4248  3280
 FNDPRT  Abs  118170 #1CD9A -  1091   461
 FPOLLj  Ext                 -  1653
 FROMDT  Ext                 -   313
 FTYP30  Abs  116932 #1C8C4 -   283   277
 FTYP40  Abs  116958 #1C8DE -   292   275   279   282   289
 FTYP50  Abs  116962 #1C8E2 -   293
 FTYPER  Abs  118623 #1CF5F -  1713   355  3225  3247
 FTYPF#  Ext                 -   276  4167
 GETSIZ  Abs  118238 #1CDDE -  1250  1432  3558
 GETSOC  Abs  118214 #1CDC6 -  1207  1427  3552
 HDRH10  Abs  121482 #1DA8A -  4167  4157  4165
 HDRH40  Abs  121537 #1DAC1 -  4182  4180
 HDRH50  Abs  121547 #1DACB -  4186  4169
 HDRH60  Abs  121577 #1DAE9 -  4196  4171
 HDRHDR  Abs  121442 #1DA62 -  4152  3222  3686
 HDRSIZ  Abs      37 #00025 -    20  3053  3060
 I/OAL+  Ext                 -  1608
 I/ODAL  Ext                 -  1642
 IDIVA   Ext                 -  1341
 IMPF10  Abs  118156 #1CD8C -  1016  1013
 IMPFLD  Abs  118106 #1CD5A -   999   405
 IOAL+   Abs  118517 #1CEF5 -  1606  3646
 IOAL36  Abs  118509 #1CEED -  1603   263  3103
 LAKEYS  Ext                 -  3238
 LC2TRK  Abs  121755 #1DB9B -  4447  3265  4016
 LCTRKS  Abs  118189 #1CDAD -  1128   433   441   694
 LIF1R   Abs   18828 #0498C -    18   328
 LIF1RT  Abs      73 #00049 -    19
 LOCBIT  Abs  121694 #1DB5E -  4396  4073  4249
 MAINEN  Abs  193905 #2F571 -    13  3825
 MAK120  Abs  121310 #1D9DE -  3936  3934
 MAK125  Abs  121343 #1D9FF -  3947  3945
 MAKH20  Abs  121225 #1D989 -  3909  3905  3907
 MAKH90  Abs  121248 #1D9A0 -  3916  3920
 MAKHDR  Abs  121184 #1D960 -  3897  3121  3661
 MAXTRK  Abs  117865 #1CC69 -   689   415
 MEMCKL  Ext                 -  3054
 MESS++  Abs  118737 #1CFD1 -  1921  1781
 MESS10  Abs  118730 #1CFCA -  1919  1866
 MESS20  Abs  118791 #1D007 -  1933  1928
```

```
 MESSAG  Abs  118728 #1CFC8 -  1918  1708  1987
 MFERRj  Ext                -  1951
 MFWRN   Ext                -  1932  1937
 MOVED3  Ext                -  3451
 MPTK10  Abs  121391 #1DA2F -  4018  4023
 MPTK20  Abs  121401 #1DA39 -  4022  4020
 MPTK30  Abs  121415 #1DA47 -  4028  4026
 MVMEM+  Ext                -  3832
 NEXTST  Abs  118468 #1CEC4 -  1528  3731
 NOCOMP  Abs  121671 #1DB47 -  4342  3341  3900  3909  3912  3918
 NOKEYS  Ext                -  2002
 NOMEM   Abs  119423 #1D27F -  3088  3055  3648
 NOSCRL  Ext                -  1532
 NROOM   Abs  118615 #1CF57 -  1711   265  3088
 OBCOLL  Ext                -  3728
 OFFSET  Abs  121345 #1DA01 -  4003  3137  3168
 PLLCRD  Abs  118599 #1CF47 -  1707  3113  3551  3655
 POLL    Ext                -   351
 PREPD1  Abs  117941 #1CCB5 -   788   784
 PREPDT  Abs  117914 #1CC9A -   779   457   506   541
 PREPHD  Abs  117898 #1CC8A -   734   499   532
 PRMPT   Abs  120643 #1D743 -  3541  3539
 PROTAB  Abs  120675 #1D763 -  3553  3559  3564  3566  3569  3571  3573
=PROTCT  Abs  120597 #1D715 -  3527
 PRTABT  Abs  120650 #1D74A -  3548  3654
 PRTEXT  Abs  120653 #1D74D -  3549  3581
 PRTU05  Abs  120619 #1D72B -  3534  3529
 PRTU10  Abs  120663 #1D757 -  3551  3542
 PRTU25  Abs  120683 #1D76B -  3556  3561
 PRTU30  Abs  120693 #1D775 -  3558  3555
 PRTUNP  Abs  120622 #1D72E -  3535  3553  3557
 R1D037  Abs  120296 #1D5E8 -  3366  3193  3205
 R1TODO  Abs  118206 #1CDBE -  1168   335  3065
 R<RST2  Ext                -  1656
 RALIGN  Abs  118860 #1D04C -  1986  3108  3296
 RC01    Ext                -   314
 RD8S    Abs  119321 #1D219 -  2874  3898  3901  3910  3916  3928
 RD8SV   Abs  119007 #1D0DF -  2288   557   929   943
 RDABT!  Abs  119491 #1D2C3 -  3110  3294
 RDS049  Abs  118396 #1CE7C -  1431  1435
 RDS050  Abs  118400 #1CE80 -  1432  1430
 RDSOC   Abs  118384 #1CE70 -  1427   482   529  3116  3301  3656
 RDYT10  Abs  118639 #1CF6F -  1764  1825  3683
 RDYTRK  Abs  118631 #1CF67 -  1762   481   528  3298
 READ8   Abs  118991 #1D0CF -  2241  1209
 READ8S  Abs  119004 #1D0DC -  2287  1250  1437  2874  3153  3307  3316  3326
                               3332  3339  3342
 READCS  Abs  119030 #1D0F6 -  2378   492  3162
 READFL  Abs  118970 #1D0BA -  2152   531   540  1208  1436  3118  3136  3303
                               3658
 REN180  Ext                -  1529
 RST2<R  Ext                -  1660
 RTNAB+  Abs  118460 #1CEBC -  1526  3550
 RTNABT  Abs  118457 #1CEB9 -  1525   480   527  3111  3295
 RTNCC   Ext                -  3526
```

```
 RTODP   Abs  118200 #1CDB8 -  1166  268  293  315  420  567  779 3123
                               3186 3282
 RWERR   Abs  118607 #1CF4F -  1709 3182 3364 3946
 SCRLLR  Ext                -  1990
 SCRTCH  Abs  194817 #2F901 -    13 1645 1939
 SETBIT  Abs  121428 #1DA54 -  4073 3189
 STO1    Ext                -   301
=STDRG?  Abs  118361 #1CE59 -  1375  278 4170
 STOPDC  Ext                -  3525
 STR1-2  Abs  118545 #1CF11 -  1645 1768 1919
 SWPBYT  Ext                -  3402
 SXM     Abs  121575 #1DAE7 -  4195 4181
 Sflag?  Ext                -  1927
 TEST    Abs       1 #00001 -    11 1485 2470 2523 2584 2681
 TIMER1  Abs  189432 #2E3F8 -    13 2779 2801
 TOCA10  Abs  117956 #1CCC4 -   851  856
 TOCA20  Abs  117977 #1CCD9 -   858  852
 TOCA30  Abs  118003 #1CCF3 -   866  863  941  953 1005
 TOCARD  Abs  117945 #1CCB9 -   847  500  507
 TODT    Ext                -   303
=TRKDON  Abs  118700 #1CFAC -  1823  566 3277
 TRKSIZ  Abs     650 #0028A -   258 1129 1252 3089 4448
 TUSLO?  Abs  119242 #1D1CA -  2819 2829
 UNKCD   Abs  118591 #1CF3F -  1705 1431 3556 3908
 UNPRDC  Ext                -  3531
=UNPROT  Abs  120616 #1D728 -  3533
 UNPRTP  Ext                -  3532
 UPDCKP  Abs  121683 #1DB53 -  4346 3927
 VFY000  Abs  117687 #1CBB7 -   524  530  534  543  558
 VFY010  Abs  117702 #1CBC6 -   528  526
 VFY035  Abs  117733 #1CBE5 -   537  546  560  563
 VFY040  Abs  117745 #1CBF1 -   540  536
 VFY060  Abs  117768 #1CC08 -   547  545
 VFY070  Abs  117788 #1CC1C -   554  564
 VFY080  Abs  117818 #1CC3A -   565  548  555
 VFYC10  Abs  118019 #1CD03 -   926  937
 VFYC20  Abs  118055 #1CD27 -   939  927
 VFYC30  Abs  118100 #1CD54 -   954  932  946
 VFYCRD  Abs  118005 #1CCF5 -   921  533  542
 VFYER!  Abs  119281 #1D1F1 -  2830 2827
 VFYER?  Abs  119252 #1D1D4 -  2822 2817
 WAIT    Abs  119108 #1D144 -  2777 2475 2528 2589
 WAITCC  Abs  119216 #1D1B0 -  2810 2800
 WAITCS  Abs  119076 #1D124 -  2582 2379
 WAITDL  Abs  119127 #1D157 -  2781 2787
 WAITER  Abs  119210 #1D1AA -  2808 2815 2820
 WAITM+  Abs  119096 #1D138 -  2632  522 3572
 WAITMT  Abs  119102 #1D13E -  2680 2192 2632
 WAITRQ  Abs  119056 #1D110 -  2469 2242
 WAITRS  Abs  119066 #1D11A -  2522 2288 2330 2420
 WIPOUT  Ext                -  3785
 WRIT4S  Abs  119043 #1D103 -  2419 3570
 WRIT8S  Abs  119017 #1D0E9 -  2329  518  855  865 2867 2869 3568
 WRITCP  Abs  118929 #1D091 -  2071 2116 2153 2194
 WRITE   Abs  118926 #1D08E -  2070  494 3562
```

```
 WRITFL  Abs  118977 #1D0C1 -  2191   497   504  3565
 WRMSG   Abs  118712 #1CFB8 -  1862   488   538  1706  1710  2809  3132  3314
 WRT2-0  Abs  119286 #1D1F6 -  2866   495   502  3563
 WTLOOP  Abs  119151 #1D16F -  2789  2804
 YMDDAY  Ext                -   310
 YMDHO1  Ext                -  3708
 aslw5   Abs  119307 #1D20B -  2872   376   454  3081  3094  3096
 asrw5   Abs  119314 #1D212 -  2873   446   786  3255
 atnclr  Ext                -  1923
 bCARD   Abs    2055 #00807 -    15  1607  1641
 crdfnm  Abs  119928 #1D478 -  3248  3260
=crlfnd  Abs  118478 #1CECE -  1531   429  1481  1526
=csrw5   Abs  119300 #1D204 -  2871   410   510   550  3097  3147  3781  3784
                           -  4410
 eCALGN  Ext                -  3651
 eDVCNF  Ext                -  1479
 eF2BIG  Ext                -   370
 eFEXST  Ext                -  3211
 eFPROT  Ext                -  3229
 eFTYPE  Ext                -  1713
 eLOBAT  Ext                -  1930
 eMEM    Ext                -  1711
 eNOTIN  Ext                -  3313
 ePLLC   Ext                -  1707
 ePLLC#  Ext                -  1762
 ePROTD  Ext                -   487
 eRALGN  Ext                -  1986
 eRWERR  Ext                -  1709  2805
 eTRKDN  Ext                -  1823
 eTRKOF  Ext                -  3681
 eTUFAS  Ext                -  2812
 eTUSLO  Ext                -  2819
 eUALGN  Ext                -  3537
 eUNKCD  Ext                -  1705
 eVALGN  Ext                -   524
 eVFYER  Ext                -   537
 eWALGN  Ext                -   477
 eWRGNM  Ext                -  3131
 errout  Abs  118841 #1D039 -  1947   371  1480  1712  1714  3212  3230
 fKEY    Abs   57868 #0E20C -    15  3242
 fTMOUT  Abs       3 #00003 -    17
 f1BAT   Abs     -61 #FFFC3 -    15  1926
 fpoll   Abs  118571 #1CF2B -  1653   475  3275  3290
=i/odal  Abs  118538 #1CF0A -  1642
 idiva   Abs  118354 #1CE52 -  1341   438   695  3090  3266
 k#ATTN  Abs      43 #0002B -    14  1995
 k#EOL   Abs      38 #00026 -    14  1993
 k#OFF   Abs      99 #00063 -    14  1997
 mvmem+  Abs  121175 #1D957 -  3831  3469
 nokeys  Abs  118913 #1D081 -  2002  1988  2004
 noscrl  Abs  118485 #1CED5 -  1532   430  1527
 pCRDAB  Abs      51 #00033 -    15  3291
 pRCRD   Abs      52 #00034 -    15  3276
 pWCRD   Abs      53 #00035 -    15   476
 pWCRD8  Abs      36 #00024 -    15   352
```

```
=r<rst2  Abs  118577 #1CF31 -  1655
=rst2<r  Abs  118584 #1CF38 -  1659
=rstlvl  Abs  118584 #1CF38 -  1660  1938  1991
=savlvl  Abs  118577 #1CF31 -  1656  1924  1989
 space1  Abs  120752 #1D7B0 -  3582
 space2  Abs  120584 #1D708 -  3471
```

Input Parameters

  Source file name is MN&CD::MS

  Listing file name is MN/CD:TI:ML::-1

  Object file name is MN%CD:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
1                       TITLE  Zero File - End of chain
2 00000 00              NIBHEX 00
3 00002                 END
```

Input Parameters

  Source file name is JP&ZER::MS

  Listing file name is JP/ZER:TI:ML::-1

  Object file name is JP%ZER:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *    TTTTT  III   &     RRRR   EEEEE  V   V
 2            *      T     I   & &    R   R  E      V   V
 3            *      T     I   & &    R   R  E      V   V
 4            *      T     I    &     RRRR   EEEE    V V
 5            *      T     I   & & &  R R    E       V V
 6            *      T     I   &  &   R  R   E        V
 7            *      T    III   && &  R   R  EEEEE    V
 8
 9                      TITLE  HP-71 Revision Number
10 00000 24             NIBASC \B\
11 00002                END
```

Input Parameters

  Source file name is TI&REV::MS

  Listing file name is TI/REV:TI:ML::-1

  Object file name is TI%REV:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                *    TTTTT  III   &     FFFFF  X   X     4
 2                *      T     I   & &    F      X   X    44
 3                *      T     I   & &    F       X X    4 4
 4                *      T     I    &     FFFF     X    4  4
 5                *      T     I   & & &  F       X X   44444
 6                *      T     I   &  &   F      X   X     4
 7                *      T    III   && &  F      X   X     4
 8                *
 9                       TITLE ROM 4 Fix Module
10 1DBAA                 ABS    #1DBAA
11                *
12 1DBAA 3400 =PKFIX1 LC(5)  =CURRST
         000
13 1DBB1 134             D0=C
14 1DBB4 184             D0=D0- 5              Point to MAINST ram loc.
15 1DBB7 868             ?ST=0  8              PEEK only?
16 1DBBA 00              RTNYES
17 1DBBC 8BB             ?D>=C  A              Blk start >= CURRST ram loc?
18 1DBBF 00              RTNYES
19 1DBC1 136             CDOEX                 Address of MAINST in C(A)
20 1DBC4 134             D0=C
21                * Block start IS prior to CURRST
22 1DBC7 8BA             ?A<=C  A              Blk end <=MAINST RAM loc?
23 1DBCA 00              RTNYES
24 1DBCC 8C00            GOLONG =POKERR
         00
25                *
26                ************************************************
27                *
28                * Fix CALL/SUB bug :
29                * When the parameters in a SUB statement have duplicate variable
30                * and they are in different type of variables, it would be error
31                * the EXPEXC directly. This will leave the machine in the local
32                * environment and the CNTADR contains garbage.
33                * The way to fix this problem is to check the existence of the va
34                * before call EXPEXC.
35                *
36 1DBD2 136  =CALFX1 CDOEX                Save D0 in RSTK
37 1DBD5 06          RSTK=C
38 1DBD7 134         D0=C
39 1DBDA 3100        LC(2)  =tDMYAR       See if is an array variable ?
40 1DBDE 14A         A=DAT0 B
41 1DBE1 966         ?A#C   B
42 1DBE4 50          GOYES  CALF10
43 1DBE6 161         D0=D0+ 2
44 1DBE9 8F00  CALF10 GOSBVL =ADDRSS       Try to find the variable
         000
45 1DBF0 07          C=RSTK                Restore D0
46 1DBF2 134         D0=C
47 1DBF5 500         RTNNC                 Return right away if var. found
48 1DBF8 8D00        GOVLNG =EXPEX-        Call EXPEXC as usual if var. not fo
         000
49                *
50 1DBFF             END
```

ADDRSS	Ext		-	44		
CALF10	Abs	121833	#1DBE9	-	44	42
=CALFX1	Abs	121810	#1DBD2	-	36	
CURRST	Ext			-	12	
EXPEX-	Ext			-	48	
=PKFIX1	Abs	121770	#1DBAA	-	12	
POKERR	Ext			-	24	
tDMYAR	Ext			-	39	

Input Parameters

  Source file name is TI&FX4::MS

  Listing file name is TI/FX4:TI:ML::-1

  Object file name is TI%FX4:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
      1                       TITLE  Timestamp File
      2 1DCCE                 ABS    #1DCCE
      3 1DCCE 4557            NIBASC \Tue Sep \
              5602
              3556
              0702
      4 1DCDE 1333            NIBASC \13, 1983\
              C202
              1393
              8333
      5 1DCEE 0202            NIBASC \  12:10 \
              1323
              A313
              0302
      6 1DCFE 16D6            NIBASC \am\
      7 1DD02 FF              NIBHEX FF
```

Input Parameters

  Source file name is TI&TS::MS

  Listing file name is TI/TS:TI:ML::-1

  Object file name is TI%TS:TI:MS::-1

                                  111111
                        0123456789012345
  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                      TITLE  Lexical Analyzer Tables--ID=01
 2 1DD04                ABS    #1DD04
 3              *        J  PPPP    &     TTTTT    A    BBBB
 4              *        J  P   P  & &      T     A A   B   B
 5              *        J  P   P  & &      T    A   A  B   B
 6              *        J  PPPP    &       T    A   A  BBBB
 7              *        J  P      & & &    T    AAAAA  B   B
 8              *     J  J  P      &  &     T    A   A  B   B
 9              *      JJJ   P      && &    T    A   A  BBBB
10              *
11              *  This file was generated on Mon Jul 18, 1983  10:29 am
12 1DD04        =xrm01s
13 1DD04 10             NIBHEX 10               Id
14 1DD06 10             CON(2)   1              Lowest Token
15 1DD08 B5             CON(2)  91              Highest Token
16 1DD0A 0000           REL(5) =MAINTS          End of lex table chain
         0
17              *
18              *  Speed Table
19 1DD0F 0              NIBHEX 0                Speed table exists
20 1DD10 000            CON(3)    0               A
21 1DD13 960            CON(3)  105               B
22 1DD16 270            CON(3)  114               C
23 1DD19 C11            CON(3)  284               D
24 1DD1C 051            CON(3)  336               E
25 1DD1F 3B1            CON(3)  435               F
26 1DD22 AE1            CON(3)  490               G
27 1DD25 602            CON(3)  518               H
28 1DD28 F02            CON(3)  527               I
29 1DD2B 594            CON(3) (TxTbEn)-(TxTbSt)  J
30 1DD2E A12            CON(3)  538               K
31 1DD31 C32            CON(3)  572               L
32 1DD34 462            CON(3)  612               M
33 1DD37 1A2            CON(3)  673               N
34 1DD3A 594            CON(3) (TxTbEn)-(TxTbSt)  O
35 1DD3D EB2            CON(3)  702               P
36 1DD40 594            CON(3) (TxTbEn)-(TxTbSt)  Q
37 1DD43 A33            CON(3)  826               R
38 1DD46 293            CON(3)  914               S
39 1DD49 704            CON(3) 1031               T
40 1DD4C 234            CON(3) 1074               U
41 1DD4F 854            CON(3) 1112               V
42 1DD52 E64            CON(3) 1134               W
43 1DD55 594            CON(3) (TxTbEn)-(TxTbSt)  X
44 1DD58 594            CON(3) (TxTbEn)-(TxTbSt)  Y
45 1DD5B A84            CON(3) 1162               Z
46 1DD5E 0              NIBHEX 0                Speed table exists
47 1DD5F 1430           CON(4) (TxTbSt)+1-(*)   Offset to text table
48 1DD63 0000           CON(4) 0                No message table
49 1DD67 0000           REL(5) =RTNSXM          Offset to poll handler
         0
```

```
50                      STITLE M a i n  T a b l e
51              *  Main Table
52 1DD6C        =xrom01
53              *
54 1DD6C 000            CON(3)    0             01  ACS
55 1DD6F 0000           REL(5) =ACOS
         0
56 1DD74 F              NIBHEX F
57              *
58 1DD75 900            CON(3)    9             02  ADDR$
59 1DD78 0000           REL(5) =ADDR$
         0
60 1DD7D F              NIBHEX F
61              *
62 1DD7E 610            CON(3)   22             03  ADJABS
63 1DD81 0000           REL(5) =ADJAAA
         0
64 1DD86 D              NIBHEX D
65              *
66 1DD87 520            CON(3)   37             04  ADJUST
67 1DD8A 0000           REL(5) =ADJNNN
         0
68 1DD8F D              NIBHEX D
69              *
70 1DD90 430            CON(3)   52             05  AF
71 1DD93 0000           REL(5) =AF
         0
72 1DD98 F              NIBHEX F
73              *
74              =xANGLE EQU    #06
75 1DD99 B30            CON(3)   59             06  ANGLE  (function and middle wo
76 1DD9C 0000           REL(5) =ANGLE
         0
77 1DDA1 F              NIBHEX F
78              *
79 1DDA2 840            CON(3)   72             07  ASN
80 1DDA5 0000           REL(5) =ASIN
         0
81 1DDAA F              NIBHEX F
82              *
83 1DDAB 150            CON(3)   81             08  ASSIGN
84 1DDAE 0000           REL(5) =ASSIGN
         0
85 1DDB3 D              NIBHEX D
86              *
87 1DDB4 060            CON(3)   96             09  ATN
88 1DDB7 0000           REL(5) =ATAN
         0
89 1DDBC F              NIBHEX F
90              *
91 1DDBD 960            CON(3)  105             0A  BYE
92 1DDC0 0000           REL(5) =BYE
         0
93 1DDC5 D              NIBHEX D
94              *
```

```
 95 1DDC6 270          CON(3)  114          0B  CAT$
 96 1DDC9 0000         REL(5) =CAT$
          0
 97 1DDCE F            NIBHEX F
 98              *
 99 1DDCF EF3          CON(3) 1022          0C  STD
100 1DDD2 0000         REL(5) =STD
          0
101 1DDD7 D            NIBHEX D
102             *
103 1DDD8 3B1          CON(3)  435          0D  FIX
104 1DDDB 0000         REL(5) =DSPF
          0
105 1DDE0 D            NIBHEX D
106             *
107 1DDE1 293          CON(3)  914          0E  SCI
108 1DDE4 0000         REL(5) =DSPF
          0
109 1DDE9 D            NIBHEX D
110             *
111 1DDEA 161          CON(3)  353          0F  ENG
112 1DDED 0000         REL(5) =DSPF
          0
113 1DDF2 D            NIBHEX D
114             *
115 1DDF3 D90          CON(3)  157          10  CHARSET
116 1DDF6 0000         REL(5) =CHARST
          0
117 1DDFB D            NIBHEX D
118             *
119 1DDFC D70          CON(3)  125          11  CHAIN
120 1DDFF 0000         REL(5) =CHAIN
          0
121 1DE04 D            NIBHEX D
122             *
123 1DE05 A80          CON(3)  138          12  CHARSET$
124 1DE08 0000         REL(5) =CHRST$
          0
125 1DE0D F            NIBHEX F
126             *
127 1DE0E EA0          CON(3)  174          13  CLAIM (PORT)
128 1DE11 0000         REL(5) =NASSAU
          0
129 1DE16 7            NIBHEX 7
130             *
131 1DE17 BB0          CON(3)  187          14  CLASS
132 1DE1A 0000         REL(5) =CLASS
          0
133 1DE1F F            NIBHEX F
134             *
135             =xCLOCK EQU    #15
136 1DE20 8C0          CON(3)  200          15  (RESET) CLOCK
137 1DE23 0000         NIBHEX 00000
          0
138 1DE28 0            NIBHEX 0
```

```
139                *
140 1DE29 5D0          CON(3)  213          16  CLSTAT
141 1DE2C 0000         REL(5) =CLSTAT
            0
142 1DE31 D            NIBHEX D
143                *
144 1DE32 4E0          CON(3)  228          17  CONTRAST
145 1DE35 0000         REL(5) =CNTRST
            0
146 1DE3A D            NIBHEX D
147                *
148 1DE3B 7F0          CON(3)  247          18  CONT
149 1DE3E 0000         REL(5) =CONT
            0
150 1DE43 7            NIBHEX 7
151                *
152 1DE44 201          CON(3)  258          19  CORR
153 1DE47 0000         REL(5) =CORR
            0
154 1DE4C F            NIBHEX F
155                *
156 1DE4D 6D2          CON(3)  726          1A  PLIST
157 1DE50 0000         REL(5) =PLIST
            0
158 1DE55 D            NIBHEX D
159                *
160 1DE56 D01          CON(3)  269          1B  CREATE
161 1DE59 0000         REL(5) =CREATE
            0
162 1DE5E D            NIBHEX D
163                *
164              =xZERO EQU    #1C
165 1DE5F A84          CON(3) 1162          1C  ZERO
166 1DE62 0000         NIBHEX 00000
            0
167 1DE67 0            NIBHEX 0
168                *
169 1DE68 C11          CON(3)  284          1D  DEFAULT
170 1DE6B 0000         REL(5) =DEFALT
            0
171 1DE70 D            NIBHEX D
172                *
173 1DE71 A31          CON(3)  314          1E  DROP
174 1DE74 0000         REL(5) =DROP
            0
175 1DE79 D            NIBHEX D
176                *
177 1DE7A 541          CON(3)  325          1F  DTH$
178 1DE7D 0000         REL(5) =HEX$
            0
179 1DE82 F            NIBHEX F
180                *
181 1DE83 051          CON(3)  336          20  ENDLINE
182 1DE86 0000         REL(5) =ENDLIN
            0
```

```
183 1DE8B D           NIBHEX D
184           *
185 1DE8C A61         CON(3)  362          21  ERRM$
186 1DE8F 0000        REL(5) =ERRM$
          0
187 1DE94 F           NIBHEX F
188           *
189 1DE95 364         CON(3) 1123          22  VER$
190 1DE98 0000        REL(5) =VER$
          0
191 1DE9D F           NIBHEX F
192           *
193 1DE9E 771         CON(3)  375          23  EXACT
194 1DEA1 0000        REL(5) =EXACTT
          0
195 1DEA6 D           NIBHEX D
196           *
197 1DEA7 481         CON(3)  388          24  EXPM1
198 1DEAA 0000        REL(5) =EXPM1
          0
199 1DEAF F           NIBHEX F
200           *
201 1DEB0 191         CON(3)  401          25  EXPONENT
202 1DEB3 0000        REL(5) =EXPON
          0
203 1DEB8 F           NIBHEX F
204           *
205           =xEXTND EQU    #26
206 1DEB9 4A1         CON(3)  420          26  EXTEND
207 1DEBC 0000        NIBHEX 00000
          0
208 1DEC1 0           NIBHEX 0
209           *
210 1DEC2 CB1         CON(3)  444          27  FLAG
211 1DEC5 0000        REL(5) =FLAG
          0
212 1DECA F           NIBHEX F
213           *
214 1DECB 7C1         CON(3)  455          28  FLOOR (Same as INT)
215 1DECE 0000        REL(5) =INT
          0
216 1DED3 F           NIBHEX F
217           *
218           =xFLOW  EQU    #29
219 1DED4 4D1         CON(3)  468          29  (TRACE) FLOW
220 1DED7 0000        NIBHEX 00000
          0
221 1DEDC 0           NIBHEX 0
222           *
223 1DEDD FD1         CON(3)  479          2A  FREE (PORT)
224 1DEE0 0000        REL(5) =FRPORT
          0
225 1DEE5 7           NIBHEX 7
226           *
227 1DEE6 9F1         CON(3)  505          2B  GDISP
```

```
228 1DEE9 0000          REL(5) =GDISP
          0
229 1DEEE D             NIBHEX D
230               *
231 1DEEF AE1           CON(3)  490             2C  GDISP$
232 1DEF2 0000          REL(5) =GDISP$
          0
233 1DEF7 F             NIBHEX F
234               *
235 1DEF8 602           CON(3)  518             2D  HTD
236 1DEFB 0000          REL(5) =HXDEC
          0
237 1DF00 F             NIBHEX F
238               *
239               =xINTO  EQU    #2E
240 1DF01 F02           CON(3)  527             2E  INTO
241 1DF04 0000          NIBHEX 00000
          0
242 1DF09 0             NIBHEX 0
243               *
244 1DF0A A12           CON(3)  538             2F  KEYDEF$
245 1DF0D 0000          REL(5) =KEYDEF
          0
246 1DF12 F             NIBHEX F
247               *
248 1DF13 B22           CON(3)  555             30  KEYDOWN
249 1DF16 0000          REL(5) =KEYDWN
          0
250 1DF1B F             NIBHEX F
251               *
252 1DF1C C32           CON(3)  572             31  LC
253 1DF1F 0000          REL(5) =FLIP
          0
254 1DF24 D             NIBHEX D
255               *
256 1DF25 342           CON(3)  579             32  LGT
257 1DF28 0000          REL(5) =LOG10
          0
258 1DF2D F             NIBHEX F
259               *
260 1DF2E C42           CON(3)  588             33  LOCK
261 1DF31 0000          REL(5) =LOCK
          0
262 1DF36 D             NIBHEX D
263               *
264 1DF37 752           CON(3)  599             34  LOGP1
265 1DF3A 0000          REL(5) =LOGP1
          0
266 1DF3F F             NIBHEX F
267               *
268 1DF40 E64           CON(3) 1134             35  WIDTH
269 1DF43 0000          REL(5) =WIDTH
          0
270 1DF48 D             NIBHEX D
271               *
```

```
272             =xMATH  EQU    #36
273 1DF49 462           CON(3)  612             36  MATH
274 1DF4C 0000          NIBHEX 00000
          0
275 1DF51 0             NIBHEX 0
276             *
277 1DF52 F62           CON(3)  623             37  MEAN (Duplicate of Built-in)
278 1DF55 0000          REL(5) =MEAN
          0
279 1DF5A F             NIBHEX F
280             *
281 1DF5B A72           CON(3)  634             38  MEM
282 1DF5E 0000          REL(5) =MEM
          0
283 1DF63 F             NIBHEX F
284             *
285 1DF64 382           CON(3)  643             39  MERGE
286 1DF67 0000          REL(5) =MERGE
          0
287 1DF6C D             NIBHEX D
288             *
289 1DF6D 092           CON(3)  656             3A  MINREAL
290 1DF70 0000          REL(5) =MINRL
          0
291 1DF75 F             NIBHEX F
292             *
293 1DF76 1A2           CON(3)  673             3B  NAN
294 1DF79 0000          REL(5) =NAN
          0
295 1DF7E F             NIBHEX F
296             *
297             =xNEAR  EQU    #3C
298 1DF7F AA2           CON(3)  682             3C  NEAR
299 1DF82 0000          NIBHEX 00000
          0
300 1DF87 0             NIBHEX 0
301             *
302             =xNEG   EQU    #3D
303 1DF88 5B2           CON(3)  693             3D  NEG
304 1DF8B 0000          NIBHEX 00000
          0
305 1DF90 0             NIBHEX 0
306             *
307             =xPCRD  EQU    #3E
308 1DF91 EB2           CON(3)  702             3E  PCRD
309 1DF94 0000          NIBHEX 00000
          0
310 1DF99 0             NIBHEX 0
311             *
312 1DF9A 9C2           CON(3)  713             3F  PEEK$
313 1DF9D 0000          REL(5) =PEEK$
          0
314 1DFA2 F             NIBHEX F
315             *
316 1DFA3 3E2           CON(3)  739             40  POKE
```

```
317 1DFA6 0000          REL(5) =POKE
          0
318 1DFAB D             NIBHEX D
319             *
320 1DFAC EE2           CON(3)  750             41  POP
321 1DFAF 0000          REL(5) =POP
          0
322 1DFB4 D             NIBHEX D
323             *
324             =xPOS   EQU    #42
325 1DFB5 7F2           CON(3)  759             42  POS
326 1DFB8 0000          REL(5) =POS
          0
327 1DFBD F             NIBHEX F
328             *
329 1DFBE 003           CON(3)  768             43  PRIVATE
330 1DFC1 0000          REL(5) =PRIVAT
          0
331 1DFC6 D             NIBHEX D
332             *
333 1DFC7 113           CON(3)  785             44  PROTECT
334 1DFCA 0000          REL(5) =PROTCT
          0
335 1DFCF D             NIBHEX D
336             *
337 1DFD0 223           CON(3)  802             45  PUT
338 1DFD3 0000          REL(5) =PUT
          0
339 1DFD8 D             NIBHEX D
340             *
341 1DFD9 B23           CON(3)  811             46  PWIDTH
342 1DFDC 0000          REL(5) =PWIDTX
          0
343 1DFE1 D             NIBHEX D
344             *
345 1DFE2 A33           CON(3)  826             47  RANDOMIZ(E)
346 1DFE5 0000          REL(5) =RANDOM
          0
347 1DFEA D             NIBHEX D
348             *
349 1DFEB D43           CON(3)  845             48  RED
350 1DFEE 0000          REL(5) =RED
          0
351 1DFF3 F             NIBHEX F
352             *
353 1DFF4 653           CON(3)  854             49  RENAME
354 1DFF7 0000          REL(5) =RENAME
          0
355 1DFFC D             NIBHEX D
356             *
357 1DFFD 563           CON(3)  869             4A  RENUMBER
358 1E000 0000          REL(5) =RENUM
          0
359 1E005 D             NIBHEX D
360             *
```

```
  361 1E006 873          CON(3)  888          4B  RESET [CLOCK]
  362 1E009 0000         REL(5) =RESET
            0
  363 1E00E D            NIBHEX D
  364             *
  365             =xROUND EQU    #4C
  366 1E00F 583          CON(3)  901          4C  ROUND
  367 1E012 0000         NIBHEX 00000
            0
  368 1E017 0            NIBHEX 0
  369             *
  370 1E018 B93          CON(3)  923          4D  SDEV (Duplicate of Built-in)
  371 1E01B 0000         REL(5) =SDEV
            0
  372 1E020 F            NIBHEX F
  373             *
  374 1E021 B74          CON(3) 1147          4E  WINDOW
  375 1E024 0000         REL(5) =WINDOW
            0
  376 1E029 D            NIBHEX D
  377             *
  378 1E02A 6A3          CON(3)  934          4F  SECURE
  379 1E02D 0000         REL(5) =SECURE
            0
  380 1E032 D            NIBHEX D
  381             *
  382 1E033 D21          CON(3)  301          50  DISP$
  383 1E036 0000         REL(5) =DSP$
            0
  384 1E03B F            NIBHEX F
  385             *
  386 1E03C 5B3          CON(3)  949          51  SETDATE
  387 1E03F 0000         REL(5) =SETDAT
            0
  388 1E044 D            NIBHEX D
  389             *
  390 1E045 6C3          CON(3)  966          52  SETTIME
  391 1E048 0000         REL(5) =SETTIM
            0
  392 1E04D D            NIBHEX D
  393             *
  394 1E04E 7D3          CON(3)  983          53  SHOW (PORT)
  395 1E051 0000         REL(5) =SHOW
            0
  396 1E056 7            NIBHEX 7
  397             *
  398 1E057 2E3          CON(3)  994          54  SQRT
  399 1E05A 0000         REL(5) =SQR
            0
  400 1E05F F            NIBHEX F
  401             *
  402 1E060 DE3          CON(3) 1005          55  STARTUP
  403 1E063 0000         REL(5) =STRTUP
            0
  404 1E068 D            NIBHEX D
```

```
405                *
406 1E069 704           CON(3) 1031          56  TOTAL
407 1E06C 0000          REL(5) =TOTAL
          0
408 1E071 F             NIBHEX F
409                *
410 1E072 414           CON(3) 1044          57  TRANSFORM
411 1E075 0000          REL(5) =TRSFMX
          0
412 1E07A D             NIBHEX D
413                *
414 1E07B 724           CON(3) 1063          58  TRAP
415 1E07E 0000          REL(5) =TRAP
          0
416 1E083 F             NIBHEX F
417                *
418 1E084 234           CON(3) 1074          59  UNPROTEC(T)
419 1E087 0000          REL(5) =UNPROT
          0
420 1E08C D             NIBHEX D
421                *
422 1E08D 544           CON(3) 1093          5A  UNSECURE
423 1E090 0000          REL(5) =UNSECR
          0
424 1E095 D             NIBHEX D
425                *
426                =xVARS  EQU    #5B
427 1E096 854           CON(3) 1112          5B  (TRACE) VARS
428 1E099 0000          NIBHEX 00000
          0
429 1E09E 0             NIBHEX 0
```

```
430                     STITLE T e x t  T a b l e
431             *  Text Table
432 1E09F        TxTbSt                       Text table start
433             *
434 1E09F 5             NIBHEX 5             ACS
435 1E0A0 1434          NIBASC \ACS\
          35
436 1E0A6 10            NIBHEX 10
437             *
438 1E0A8 9             NIBHEX 9             ADDR$
439 1E0A9 1444          NIBASC \ADDR$\
          4425
          42
440 1E0B3 20            NIBHEX 20
441             *
442 1E0B5 B             NIBHEX B             ADJABS
443 1E0B6 1444          NIBASC \ADJABS\
          A414
          2435
444 1E0C2 30            NIBHEX 30
445             *
446 1E0C4 B             NIBHEX B             ADJUST
447 1E0C5 1444          NIBASC \ADJUST\
          A455
          3545
448 1E0D1 40            NIBHEX 40
449             *
450 1E0D3 3             NIBHEX 3             AF
451 1E0D4 1464          NIBASC \AF\
452 1E0D8 50            NIBHEX 50
453             *
454 1E0DA 9             NIBHEX 9             ANGLE  (function and
455 1E0DB 14E4          NIBASC \ANGLE\
          74C4
          54
456 1E0E5 60            NIBHEX 60
457             *
458 1E0E7 5             NIBHEX 5             ASN
459 1E0E8 1435          NIBASC \ASN\
          E4
460 1E0EE 70            NIBHEX 70
461             *
462 1E0F0 B             NIBHEX B             ASSIGN
463 1E0F1 1435          NIBASC \ASSIGN\
          3594
          74E4
464 1E0FD 80            NIBHEX 80
465             *
466 1E0FF 5             NIBHEX 5             ATN
467 1E100 1445          NIBASC \ATN\
          E4
468 1E106 90            NIBHEX 90
469             *
470 1E108 5             NIBHEX 5             BYE
471 1E109 2495          NIBASC \BYE\
```

```
            54
    472 1E10F A0            NIBHEX A0
    473                *
    474 1E111 7             NIBHEX 7                CAT$
    475 1E112 3414          NIBASC \CAT$\
              4542
    476 1E11A B0            NIBHEX B0
    477                *
    478 1E11C 9             NIBHEX 9                CHAIN
    479 1E11D 3484          NIBASC \CHAIN\
              1494
              E4
    480 1E127 11            NIBHEX 11
    481                *
    482 1E129 F             NIBHEX F                CHARSET$
    483 1E12A 3484          NIBASC \CHARSET$\
              1425
              3554
              4542
    484 1E13A 21            NIBHEX 21
    485                *
    486 1E13C D             NIBHEX D                CHARSET
    487 1E13D 3484          NIBASC \CHARSET\
              1425
              3554
              45
    488 1E14B 01            NIBHEX 01
    489                *
    490 1E14D 9             NIBHEX 9                CLAIM (PORT)
    491 1E14E 34C4          NIBASC \CLAIM\
              1494
              D4
    492 1E158 31            NIBHEX 31
    493                *
    494 1E15A 9             NIBHEX 9                CLASS
    495 1E15B 34C4          NIBASC \CLASS\
              1435
              35
    496 1E165 41            NIBHEX 41
    497                *
    498 1E167 9             NIBHEX 9                (RESET) CLOCK
    499 1E168 34C4          NIBASC \CLOCK\
              F434
              B4
    500 1E172 51            NIBHEX 51
    501                *
    502 1E174 B             NIBHEX B                CLSTAT
    503 1E175 34C4          NIBASC \CLSTAT\
              3545
              1445
    504 1E181 61            NIBHEX 61
    505                *
    506 1E183 F             NIBHEX F                CONTRAST
    507 1E184 34F4          NIBASC \CONTRAST\
              E445
```

```
          2514
          3545
  508 1E194 71          NIBHEX 71
  509           *
  510 1E196 7           NIBHEX 7              CONT
  511 1E197 34F4        NIBASC \CONT\
          E445
  512 1E19F 81          NIBHEX 81
  513           *
  514 1E1A1 7           NIBHEX 7              CORR
  515 1E1A2 34F4        NIBASC \CORR\
          2525
  516 1E1AA 91          NIBHEX 91
  517           *
  518 1E1AC B           NIBHEX B              CREATE
  519 1E1AD 3425        NIBASC \CREATE\
          5414
          4554
  520 1E1B9 B1          NIBHEX B1
  521           *
  522 1E1BB D           NIBHEX D              DEFAULT
  523 1E1BC 4454        NIBASC \DEFAULT\
          6414
          55C4
          45
  524 1E1CA D1          NIBHEX D1
  525           *
  526 1E1CC 9           NIBHEX 9              DISP$
  527 1E1CD 4494        NIBASC \DISP$\
          3505
          42
  528 1E1D7 05          NIBHEX 05
  529           *
  530 1E1D9 7           NIBHEX 7              DROP
  531 1E1DA 4425        NIBASC \DROP\
          F405
  532 1E1E2 E1          NIBHEX E1
  533           *
  534 1E1E4 7           NIBHEX 7              DTH$
  535 1E1E5 4445        NIBASC \DTH$\
          8442
  536 1E1ED F1          NIBHEX F1
  537           *
  538 1E1EF D           NIBHEX D              ENDLINE
  539 1E1F0 54E4        NIBASC \ENDLINE\
          44C4
          94E4
          54
  540 1E1FE 02          NIBHEX 02
  541           *
  542 1E200 5           NIBHEX 5              ENG
  543 1E201 54E4        NIBASC \ENG\
          74
  544 1E207 F0          NIBHEX F0
  545           *
```

```
546 1E209 9           NIBHEX 9              ERRM$
547 1E20A 5425        NIBASC \ERRM$\
          25D4
          42
548 1E214 12          NIBHEX 12
549           *
550 1E216 9           NIBHEX 9              EXACT
551 1E217 5485        NIBASC \EXACT\
          1434
          45
552 1E221 32          NIBHEX 32
553           *
554 1E223 9           NIBHEX 9              EXPM1
555 1E224 5485        NIBASC \EXPM1\
          05D4
          13
556 1E22E 42          NIBHEX 42
557           *
558 1E230 F           NIBHEX F              EXPONENT
559 1E231 5485        NIBASC \EXPONENT\
          05F4
          E454
          E445
560 1E241 52          NIBHEX 52
561           *
562 1E243 B           NIBHEX B              EXTEND
563 1E244 5485        NIBASC \EXTEND\
          4554
          E444
564 1E250 62          NIBHEX 62
565           *
566 1E252 5           NIBHEX 5              FIX
567 1E253 6494        NIBASC \FIX\
          85
568 1E259 D0          NIBHEX D0
569           *
570 1E25B 7           NIBHEX 7              FLAG
571 1E25C 64C4        NIBASC \FLAG\
          1474
572 1E264 72          NIBHEX 72
573           *
574 1E266 9           NIBHEX 9              FLOOR (Same as INT)
575 1E267 64C4        NIBASC \FLOOR\
          F4F4
          25
576 1E271 82          NIBHEX 82
577           *
578 1E273 7           NIBHEX 7              (TRACE) FLOW
579 1E274 64C4        NIBASC \FLOW\
          F475
580 1E27C 92          NIBHEX 92
581           *
582 1E27E 7           NIBHEX 7              FREE (PORT)
583 1E27F 6425        NIBASC \FREE\
          5454
```

```
  584 1E287 A2          NIBHEX A2
  585           *
  586 1E289 B           NIBHEX B              GDISP$
  587 1E28A 7444        NIBASC \GDISP$\
            9435
            0542
  588 1E296 C2          NIBHEX C2
  589           *
  590 1E298 9           NIBHEX 9              GDISP
  591 1E299 7444        NIBASC \GDISP\
            9435
            05
  592 1E2A3 B2          NIBHEX B2
  593           *
  594 1E2A5 5           NIBHEX 5              HTD
  595 1E2A6 8445        NIBASC \HTD\
            44
  596 1E2AC D2          NIBHEX D2
  597           *
  598 1E2AE 7           NIBHEX 7              INTO
  599 1E2AF 94E4        NIBASC \INTO\
            45F4
  600 1E2B7 E2          NIBHEX E2
  601           *
  602 1E2B9 D           NIBHEX D              KEYDEF$
  603 1E2BA B454        NIBASC \KEYDEF$\
            9544
            5464
            42
  604 1E2C8 F2          NIBHEX F2
  605           *
  606 1E2CA D           NIBHEX D              KEYDOWN
  607 1E2CB B454        NIBASC \KEYDOWN\
            9544
            F475
            E4
  608 1E2D9 03          NIBHEX 03
  609           *
  610 1E2DB 3           NIBHEX 3              LC
  611 1E2DC C434        NIBASC \LC\
  612 1E2E0 13          NIBHEX 13
  613           *
  614 1E2E2 5           NIBHEX 5              LGT
  615 1E2E3 C474        NIBASC \LGT\
            45
  616 1E2E9 23          NIBHEX 23
  617           *
  618 1E2EB 7           NIBHEX 7              LOCK
  619 1E2EC C4F4        NIBASC \LOCK\
            34B4
  620 1E2F4 33          NIBHEX 33
  621           *
  622 1E2F6 9           NIBHEX 9              LOGP1
  623 1E2F7 C4F4        NIBASC \LOGP1\
            7405
```

```
            13
624 1E301 43          NIBHEX 43
625             *
626 1E303 7           NIBHEX 7              MATH
627 1E304 D414        NIBASC \MATH\
            4584
628 1E30C 63          NIBHEX 63
629             *
630 1E30E 7           NIBHEX 7              MEAN (Duplicate of B
631 1E30F D454        NIBASC \MEAN\
            14E4
632 1E317 73          NIBHEX 73
633             *
634 1E319 5           NIBHEX 5              MEM
635 1E31A D454        NIBASC \MEM\
            D4
636 1E320 83          NIBHEX 83
637             *
638 1E322 9           NIBHEX 9              MERGE
639 1E323 D454        NIBASC \MERGE\
            2574
            54
640 1E32D 93          NIBHEX 93
641             *
642 1E32F D           NIBHEX D              MINREAL
643 1E330 D494        NIBASC \MINREAL\
            E425
            5414
            C4
644 1E33E A3          NIBHEX A3
645             *
646 1E340 5           NIBHEX 5              NAN
647 1E341 E414        NIBASC \NAN\
            E4
648 1E347 B3          NIBHEX B3
649             *
650 1E349 7           NIBHEX 7              NEAR
651 1E34A E454        NIBASC \NEAR\
            1425
652 1E352 C3          NIBHEX C3
653             *
654 1E354 5           NIBHEX 5              NEG
655 1E355 E454        NIBASC \NEG\
            74
656 1E35B D3          NIBHEX D3
657             *
658 1E35D 7           NIBHEX 7              PCRD
659 1E35E 0534        NIBASC \PCRD\
            2544
660 1E366 E3          NIBHEX E3
661             *
662 1E368 9           NIBHEX 9              PEEK$
663 1E369 0554        NIBASC \PEEK$\
            54B4
            42
```

```
    664 1E373 F3          NIBHEX F3
    665             *
    666 1E375 9           NIBHEX 9              PLIST
    667 1E376 05C4        NIBASC \PLIST\
              9435
              45
    668 1E380 A1          NIBHEX A1
    669             *
    670 1E382 7           NIBHEX 7              POKE
    671 1E383 05F4        NIBASC \POKE\
              B454
    672 1E38B 04          NIBHEX 04
    673             *
    674 1E38D 5           NIBHEX 5              POP
    675 1E38E 05F4        NIBASC \POP\
              05
    676 1E394 14          NIBHEX 14
    677             *
    678 1E396 5           NIBHEX 5              POS
    679 1E397 05F4        NIBASC \POS\
              35
    680 1E39D 24          NIBHEX 24
    681             *
    682 1E39F D           NIBHEX D              PRIVATE
    683 1E3A0 0525        NIBASC \PRIVATE\
              9465
              1445
              54
    684 1E3AE 34          NIBHEX 34
    685             *
    686 1E3B0 D           NIBHEX D              PROTECT
    687 1E3B1 0525        NIBASC \PROTECT\
              F445
              5434
              45
    688 1E3BF 44          NIBHEX 44
    689             *
    690 1E3C1 5           NIBHEX 5              PUT
    691 1E3C2 0555        NIBASC \PUT\
              45
    692 1E3C8 54          NIBHEX 54
    693             *
    694 1E3CA B           NIBHEX B              PWIDTH
    695 1E3CB 0575        NIBASC \PWIDTH\
              9444
              4584
    696 1E3D7 64          NIBHEX 64
    697             *
    698 1E3D9 F           NIBHEX F              RANDOMIZ(E)
    699 1E3DA 2514        NIBASC \RANDOMIZ\
              E444
              F4D4
              94A5
    700 1E3EA 74          NIBHEX 74
    701             *
```

```
    702 1E3EC 5           NIBHEX 5              RED
    703 1E3ED 2554        NIBASC \RED\
              44
    704 1E3F3 84          NIBHEX 84
    705             *
    706 1E3F5 B           NIBHEX B              RENAME
    707 1E3F6 2554        NIBASC \RENAME\
              E414
              D454
    708 1E402 94          NIBHEX 94
    709             *
    710 1E404 F           NIBHEX F              RENUMBER
    711 1E405 2554        NIBASC \RENUMBER\
              E455
              D424
              5425
    712 1E415 A4          NIBHEX A4
    713             *
    714 1E417 9           NIBHEX 9              RESET [CLOCK]
    715 1E418 2554        NIBASC \RESET\
              3554
              45
    716 1E422 B4          NIBHEX B4
    717             *
    718 1E424 9           NIBHEX 9              ROUND
    719 1E425 25F4        NIBASC \ROUND\
              55E4
              44
    720 1E42F C4          NIBHEX C4
    721             *
    722 1E431 5           NIBHEX 5              SCI
    723 1E432 3534        NIBASC \SCI\
              94
    724 1E438 E0          NIBHEX E0
    725             *
    726 1E43A 7           NIBHEX 7              SDEV (Duplicate of B
    727 1E43B 3544        NIBASC \SDEV\
              5465
    728 1E443 D4          NIBHEX D4
    729             *
    730 1E445 B           NIBHEX B              SECURE
    731 1E446 3554        NIBASC \SECURE\
              3455
              2554
    732 1E452 F4          NIBHEX F4
    733             *
    734 1E454 D           NIBHEX D              SETDATE
    735 1E455 3554        NIBASC \SETDATE\
              4544
              1445
              54
    736 1E463 15          NIBHEX 15
    737             *
    738 1E465 D           NIBHEX D              SETTIME
    739 1E466 3554        NIBASC \SETTIME\
```

```
            4545
            94D4
            54
    740 1E474 25          NIBHEX 25
    741             *
    742 1E476 7           NIBHEX 7              SHOW (PORT)
    743 1E477 3584        NIBASC \SHOW\
            F475
    744 1E47F 35          NIBHEX 35
    745             *
    746 1E481 7           NIBHEX 7              SQRT
    747 1E482 3515        NIBASC \SQRT\
            2545
    748 1E48A 45          NIBHEX 45
    749             *
    750 1E48C D           NIBHEX D              STARTUP
    751 1E48D 3545        NIBASC \STARTUP\
            1425
            4555
            05
    752 1E49B 55          NIBHEX 55
    753             *
    754 1E49D 5           NIBHEX 5              STD
    755 1E49E 3545        NIBASC \STD\
            44
    756 1E4A4 C0          NIBHEX C0
    757             *
    758 1E4A6 9           NIBHEX 9              TOTAL
    759 1E4A7 45F4        NIBASC \TOTAL\
            4514
            C4
    760 1E4B1 65          NIBHEX 65
    761             *
    762 1E4B3 F           NIBHEX F              TRANSFORM
    763 1E4B4 4525        NIBASC \TRANSFOR\
            14E4
            3564
            F425
    764 1E4C4 75          NIBHEX 75
    765             *
    766 1E4C6 7           NIBHEX 7              TRAP
    767 1E4C7 4525        NIBASC \TRAP\
            1405
    768 1E4CF 85          NIBHEX 85
    769             *
    770 1E4D1 F           NIBHEX F              UNPROTEC(T)
    771 1E4D2 55E4        NIBASC \UNPROTEC\
            0525
            F445
            5434
    772 1E4E2 95          NIBHEX 95
    773             *
    774 1E4E4 F           NIBHEX F              UNSECURE
    775 1E4E5 55E4        NIBASC \UNSECURE\
            3554
```

```
            3455
            2554
    776 1E4F5 A5            NIBHEX A5
    777             *
    778 1E4F7 7             NIBHEX 7                (TRACE) VARS
    779 1E4F8 6514          NIBASC \VARS\
            2535
    780 1E500 B5            NIBHEX B5
    781             *
    782 1E502 7             NIBHEX 7                VER$
    783 1E503 6554          NIBASC \VER$\
            2542
    784 1E50B 22            NIBHEX 22
    785             *
    786 1E50D 9             NIBHEX 9                WIDTH
    787 1E50E 7594          NIBASC \WIDTH\
            4445
            84
    788 1E518 53            NIBHEX 53
    789             *
    790 1E51A B             NIBHEX B                WINDOW
    791 1E51B 7594          NIBASC \WINDOW\
            E444
            F475
    792 1E527 E4            NIBHEX E4
    793             *
    794 1E529 7             NIBHEX 7                ZERO
    795 1E52A A554          NIBASC \ZERO\
            25F4
    796 1E532 C1            NIBHEX C1
    797 1E534 1FF   TxTbEn  NIBHEX 1FF              Text termination
    798 1E537               END
```

ACOS	Ext	-	55		
ADDR$	Ext	-	59		
ADJAAA	Ext	-	63		
ADJNNN	Ext	-	67		
AF	Ext	-	71		
ANGLE	Ext	-	76		
ASIN	Ext	-	80		
ASSIGN	Ext	-	84		
ATAN	Ext	-	88		
BYE	Ext	-	92		
CAT$	Ext	-	96		
CHAIN	Ext	-	120		
CHARST	Ext	-	116		
CHRST$	Ext	-	124		
CLASS	Ext	-	132		
CLSTAT	Ext	-	141		
CNTRST	Ext	-	145		
CONT	Ext	-	149		
CORR	Ext	-	153		
CREATE	Ext	-	161		
DEFALT	Ext	-	170		
DROP	Ext	-	174		
DSP$	Ext	-	383		
DSPF	Ext	-	104	108	112
ENDLIN	Ext	-	182		
ERRM$	Ext	-	186		
EXACTT	Ext	-	194		
EXPM1	Ext	-	198		
EXPON	Ext	-	202		
FLAG	Ext	-	211		
FLIP	Ext	-	253		
FRPORT	Ext	-	224		
GDISP	Ext	-	228		
GDISP$	Ext	-	232		
HEX$	Ext	-	178		
HXDEC	Ext	-	236		
INT	Ext	-	215		
KEYDEF	Ext	-	245		
KEYDWN	Ext	-	249		
LOCK	Ext	-	261		
LOG10	Ext	-	257		
LOGP1	Ext	-	265		
MAINTS	Ext	-	16		
MEAN	Ext	-	278		
MEM	Ext	-	282		
MERGE	Ext	-	286		
MINRL	Ext	-	290		
NAN	Ext	-	294		
NASSAU	Ext	-	128		
PEEK$	Ext	-	313		
PLIST	Ext	-	157		
POKE	Ext	-	317		
POP	Ext	-	321		
POS	Ext	-	326		
PRIVAT	Ext	-	330		

PROTCT	Ext		-	334						
PUT	Ext		-	338						
PWIDTX	Ext		-	342						
RANDOM	Ext		-	346						
RED	Ext		-	350						
RENAME	Ext		-	354						
RENUM	Ext		-	358						
RESET	Ext		-	362						
RTNSXM	Ext		-	49						
SDEV	Ext		-	371						
SECURE	Ext		-	379						
SETDAT	Ext		-	387						
SETTIM	Ext		-	391						
SHOW	Ext		-	395						
SQR	Ext		-	399						
STD	Ext		-	100						
STRTUP	Ext		-	403						
TOTAL	Ext		-	407						
TRAP	Ext		-	415						
TRSFMX	Ext		-	411						
TxTbEn	Abs	124212 #1E534	-	797	29	34	36	43	44	
TxTbSt	Abs	123039 #1E09F	-	432	29	34	36	43	44	47
UNPROT	Ext		-	419						
UNSECR	Ext		-	423						
VER$	Ext		-	190						
WIDTH	Ext		-	269						
WINDOW	Ext		-	375						
=xANGLE	Abs	6 #00006	-	74						
=xCLOCK	Abs	21 #00015	-	135						
=xEXTND	Abs	38 #00026	-	205						
=xFLOW	Abs	41 #00029	-	218						
=xINTO	Abs	46 #0002E	-	239						
=xMATH	Abs	54 #00036	-	272						
=xNEAR	Abs	60 #0003C	-	297						
=xNEG	Abs	61 #0003D	-	302						
=xPCRD	Abs	62 #0003E	-	307						
=xPOS	Abs	66 #00042	-	324						
=xROUND	Abs	76 #0004C	-	365						
=xVARS	Abs	91 #0005B	-	426						
=xZERO	Abs	28 #0001C	-	164						
=xrm01s	Abs	122116 #1DD04	-	12						
=xrom01	Abs	122220 #1DD6C	-	52						

Input Parameters

  Source file name is JP&TAB::MS

  Listing file name is JP/TAB:TI:ML::-1

  Object file name is JP%TAB:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                      TITLE  Lexical Analyzer Tables--ID=00
 2 1E537                ABS    #1E537
 3              *     SSS   BBBB    &     TTTTT    A    BBBB
 4              *    S   S  B   B  & &      T     A A   B   B
 5              *    S      B   B  & &      T    A   A  B   B
 6              *     SSS   BBBB    &       T    A   A  BBBB
 7              *        S  B   B  & & &    T    AAAAA  B   B
 8              *    S   S  B   B  &  &     T    A   A  B   B
 9              *     SSS   BBBB    && &    T    A   A  BBBB
10              *
11              *  This file was generated on Thu Jul  7, 1983   3:35 pm
12 1E537        =MAINTS
13 1E537 00             NIBHEX 00               Id
14 1E539 00             CON(2)   0              Lowest Token
15 1E53B FF             CON(2) 255              Highest Token
16 1E53D 0000           NIBHEX 00000            End of lex table chain
         0
17              *
18              *  Speed Table
19 1E542 0              NIBHEX 0                 Speed table exists
20 1E543        =LXSPDT
21 1E543 000            CON(3)    0               A
22 1E546 050            CON(3)   80               B
23 1E549 660            CON(3)  102               C
24 1E54C CB0            CON(3)  188               D
25 1E54F 551            CON(3)  341               E
26 1E552 4B1            CON(3)  436               F
27 1E555 3E1            CON(3)  483               G
28 1E558 E35            CON(3) (TxTbEn)-(TxTbSt)  H
29 1E55B 202            CON(3)  514               I
30 1E55E E35            CON(3) (TxTbEn)-(TxTbSt)  J
31 1E561 662            CON(3)  614               K
32 1E564 582            CON(3)  645               L
33 1E567 5D2            CON(3)  725               M
34 1E56A 713            CON(3)  791               N
35 1E56D F33            CON(3)  831               O
36 1E570 E63            CON(3)  878               P
37 1E573 E35            CON(3) (TxTbEn)-(TxTbSt)  Q
38 1E576 4B3            CON(3)  948               R
39 1E579 134            CON(3) 1073               S
40 1E57C 6A4            CON(3) 1190               T
41 1E57F CF4            CON(3) 1276               U
42 1E582 A25            CON(3) 1322               V
43 1E585 335            CON(3) 1331               W
44 1E588 E35            CON(3) (TxTbEn)-(TxTbSt)  X
45 1E58B E35            CON(3) (TxTbEn)-(TxTbSt)  Y
46 1E58E E35            CON(3) (TxTbEn)-(TxTbSt)  Z
47 1E591 0              NIBHEX 0                Speed table exists
48 1E592 E090           CON(4) (TxTbSt)+1-(*)   Offset to text table
49 1E596 0000           CON(4) 0                No message table
50 1E59A 0000           REL(5) =RTNSXM          Offset to poll handler
         0
```

```
51                      STITLE M a i n  T a b l e
52              *  Main Table
53 1E59F        =MAINT
54              *
55 1E59F CC1            CON(3)  460             00  FN (lex only)
56 1E5A2 0000           REL(5) =FN-GO
         0
57 1E5A7 0              NIBHEX 0
58              *
59 1E5A8 000            CON(3)   0              01  Dummy Fill
60 1E5AB 0000           REL(5) =TRMNTR
         0
61 1E5B0 0              NIBHEX 0
62              *
63              =tINT12 EQU    #02
64 1E5B1 000            CON(3)   0              02  12-Digit Integer
65 1E5B4 0000           REL(5) =BLDNUM
         0
66 1E5B9 0              NIBHEX 0
67              *
68              =tINT11 EQU    #03
69 1E5BA 000            CON(3)   0              03  11-Digit Integer
70 1E5BD 0000           REL(5) =BLDNUM
         0
71 1E5C2 0              NIBHEX 0
72              *
73              =tINT10 EQU    #04
74 1E5C3 000            CON(3)   0              04  10-Digit Integer
75 1E5C6 0000           REL(5) =BLDNUM
         0
76 1E5CB 0              NIBHEX 0
77              *
78              =tINT9  EQU    #05
79 1E5CC 000            CON(3)   0              05  9-Digit Integer
80 1E5CF 0000           REL(5) =BLDNUM
         0
81 1E5D4 0              NIBHEX 0
82              *
83              =tINT8  EQU    #06
84 1E5D5 000            CON(3)   0              06  8-Digit Integer
85 1E5D8 0000           REL(5) =BLDNUM
         0
86 1E5DD 0              NIBHEX 0
87              *
88              =tINT7  EQU    #07
89 1E5DE 000            CON(3)   0              07  7-Digit Integer
90 1E5E1 0000           REL(5) =BLDNUM
         0
91 1E5E6 0              NIBHEX 0
92              *
93              =tINT6  EQU    #08
94 1E5E7 000            CON(3)   0              08  6-Digit Integer
95 1E5EA 0000           REL(5) =BLDNUM
         0
96 1E5EF 0              NIBHEX 0
```

```
 97                *
 98                =tINT5  EQU    #09
 99 1E5F0 000              CON(3)    0          09  5-Digit Integer
100 1E5F3 0000             REL(5) =BLDNUM
          0
101 1E5F8 0               NIBHEX 0
102               *
103               =tINT4  EQU    #0A
104 1E5F9 000             CON(3)    0          0A  4-Digit Integer
105 1E5FC 0000            REL(5) =BLDNUM
          0
106 1E601 0               NIBHEX 0
107               *
108               =tINT3  EQU    #0B
109 1E602 000             CON(3)    0          0B  3-Digit Integer
110 1E605 0000            REL(5) =BLDNUM
          0
111 1E60A 0               NIBHEX 0
112               *
113               =tINT2  EQU    #0C
114 1E60B 000             CON(3)    0          0C  2-Digit Integer
115 1E60E 0000            REL(5) =BLDNUM
          0
116 1E613 0               NIBHEX 0
117               *
118 1E614 000             CON(3)    0          0D  [Unused]
119 1E617 0000            REL(5) =TRMNTR
          0
120 1E61C 0               NIBHEX 0
121               *
122               =tLBLRF EQU    #0E
123 1E61D 000             CON(3)    0          0E  Label Reference
124 1E620 0000            REL(5) =TRMNTR
          0
125 1E625 0               NIBHEX 0
126               *
127               =tLINE# EQU    #0F
128 1E626 000             CON(3)    0          0F  Line Number
129 1E629 0000            REL(5) =TRMNTR
          0
130 1E62E 0               NIBHEX 0
131               *
132               =tBIG   EQU    #10
133 1E62F 000             CON(3)    0          10  Constant Too Big
134 1E632 0000            REL(5) =TRMNTR
          0
135 1E637 0               NIBHEX 0
136               *
137               =tSMALL EQU    #11
138 1E638 000             CON(3)    0          11  Constant Too Small
139 1E63B 0000            REL(5) =TRMNTR
          0
140 1E640 0               NIBHEX 0
141               *
142               =tFLT12 EQU    #12
```

```
143 1E641 000          CON(3)    0          12  12-Digit Float
144 1E644 0000         REL(5) =BLDNUM
          0
145 1E649 0            NIBHEX 0
146             *
147             =tFLT11 EQU    #13
148 1E64A 000          CON(3)    0          13  11-Digit Float
149 1E64D 0000         REL(5) =BLDNUM
          0
150 1E652 0            NIBHEX 0
151             *
152             =tFLT10 EQU    #14
153 1E653 000          CON(3)    0          14  10-Digit Float
154 1E656 0000         REL(5) =BLDNUM
          0
155 1E65B 0            NIBHEX 0
156             *
157             =tFLT9  EQU    #15
158 1E65C 000          CON(3)    0          15  9-Digit Float
159 1E65F 0000         REL(5) =BLDNUM
          0
160 1E664 0            NIBHEX 0
161             *
162             =tFLT8  EQU    #16
163 1E665 000          CON(3)    0          16  8-Digit Float
164 1E668 0000         REL(5) =BLDNUM
          0
165 1E66D 0            NIBHEX 0
166             *
167             =tFLT7  EQU    #17
168 1E66E 000          CON(3)    0          17  7-Digit Float
169 1E671 0000         REL(5) =BLDNUM
          0
170 1E676 0            NIBHEX 0
171             *
172             =tFLT6  EQU    #18
173 1E677 000          CON(3)    0          18  6-Digit Float
174 1E67A 0000         REL(5) =BLDNUM
          0
175 1E67F 0            NIBHEX 0
176             *
177             =tFLT5  EQU    #19
178 1E680 000          CON(3)    0          19  5-Digit Float
179 1E683 0000         REL(5) =BLDNUM
          0
180 1E688 0            NIBHEX 0
181             *
182             =tFLT4  EQU    #1A
183 1E689 000          CON(3)    0          1A  4-Digit Float
184 1E68C 0000         REL(5) =BLDNUM
          0
185 1E691 0            NIBHEX 0
186             *
187             =tFLT3  EQU    #1B
188 1E692 000          CON(3)    0          1B  3-Digit Float
```

```
189 1E695 0000          REL(5) =BLDNUM
          0
190 1E69A 0             NIBHEX 0
191               *
192               =tFLT2  EQU    #1C
193 1E69B 000           CON(3)    0           1C  2-Digit Float
194 1E69E 0000          REL(5) =BLDNUM
          0
195 1E6A3 0             NIBHEX 0
196               *
197               =tFLT1  EQU    #1D
198 1E6A4 000           CON(3)    0           1D  1-Digit Float
199 1E6A7 0000          REL(5) =BLDNUM
          0
200 1E6AC 0             NIBHEX 0
201               *
202 1E6AD 000           CON(3)    0           1E  [Unused]
203 1E6B0 0000          REL(5) =TRMNTR
          0
204 1E6B5 0             NIBHEX 0
205               *
206 1E6B6 000           CON(3)    0           1F  [Unused]
207 1E6B9 0000          REL(5) =TRMNTR
          0
208 1E6BE 0             NIBHEX 0
209               *
210 1E6BF 000           CON(3)    0           20  [Unused]
211 1E6C2 0000          REL(5) =TRMNTR
          0
212 1E6C7 0             NIBHEX 0
213               *
214               =a!     EQU    #21
215 1E6C8 000           CON(3)    0           21  (!)
216 1E6CB 0000          REL(5) =TRMNTR
          0
217 1E6D0 0             NIBHEX 0
218               *
219               =a"     EQU    #22
220 1E6D1 000           CON(3)    0           22  (") (String Delimiter)
221 1E6D4 0000          REL(5) =STRLIT
          0
222 1E6D9 0             NIBHEX 0
223               *
224 1E6DA 000           CON(3)    0           23  (#)
225 1E6DD 0000          REL(5) =TRMNTR
          0
226 1E6E2 0             NIBHEX 0
227               *
228               =a$     EQU    #24
229 1E6E3 000           CON(3)    0           24  ($)
230 1E6E6 0000          REL(5) =TRMNTR
          0
231 1E6EB 0             NIBHEX 0
232               *
233 1E6EC 000           CON(3)    0           25  (%)
```

```
234 1E6EF 0000          REL(5) =TRMNTR
          0
235 1E6F4 0             NIBHEX 0
236               *
237 1E6F5 000           CON(3)   0            26  (&)
238 1E6F8 0000          REL(5) =TRMNTR
          0
239 1E6FD 0             NIBHEX 0
240               *
241               =a'   EQU    #27
242 1E6FE 000           CON(3)   0            27  (') (String Delimiter)
243 1E701 0000          REL(5) =STRLIT
          0
244 1E706 0             NIBHEX 0
245               *
246 1E707 000           CON(3)   0            28  (
247 1E70A 0000          REL(5) =TRMNTR
          0
248 1E70F 0             NIBHEX 0
249               *
250 1E710 000           CON(3)   0            29  )
251 1E713 0000          REL(5) =TRMNTR
          0
252 1E718 0             NIBHEX 0
253               *
254 1E719 000           CON(3)   0            2A  (*)
255 1E71C 0000          REL(5) =TRMNTR
          0
256 1E721 0             NIBHEX 0
257               *
258 1E722 000           CON(3)   0            2B  (+)
259 1E725 0000          REL(5) =TRMNTR
          0
260 1E72A 0             NIBHEX 0
261               *
262 1E72B 000           CON(3)   0            2C  (,)
263 1E72E 0000          REL(5) =TRMNTR
          0
264 1E733 0             NIBHEX 0
265               *
266               =tSVAR EQU   #2D
267 1E734 000           CON(3)   0            2D  String Variable  (-)
268 1E737 0000          REL(5) =STRING
          0
269 1E73C 0             NIBHEX 0
270               *
271               =a.   EQU    #2E
272 1E73D 000           CON(3)   0            2E  (.)
273 1E740 0000          REL(5) =TRMNTR
          0
274 1E745 0             NIBHEX 0
275               *
276 1E746 000           CON(3)   0            2F  (/)
277 1E749 0000          REL(5) =TRMNTR
          0
```

```
278 1E74E 0           NIBHEX 0
279           *
280           =a0     EQU    #30
281 1E74F 000         CON(3)   0            30  0 (Digit)
282 1E752 0000        REL(5) =ONEDGT
          0
283 1E757 0           NIBHEX 0
284           *
285           =a1     EQU    #31
286 1E758 000         CON(3)   0            31  1 (Digit)
287 1E75B 0000        REL(5) =ONEDGT
          0
288 1E760 0           NIBHEX 0
289           *
290           =a2     EQU    #32
291 1E761 000         CON(3)   0            32  2 (Digit)
292 1E764 0000        REL(5) =ONEDGT
          0
293 1E769 0           NIBHEX 0
294           *
295           =a3     EQU    #33
296 1E76A 000         CON(3)   0            33  3 (Digit)
297 1E76D 0000        REL(5) =ONEDGT
          0
298 1E772 0           NIBHEX 0
299           *
300           =a4     EQU    #34
301 1E773 000         CON(3)   0            34  4 (Digit)
302 1E776 0000        REL(5) =ONEDGT
          0
303 1E77B 0           NIBHEX 0
304           *
305           =a5     EQU    #35
306 1E77C 000         CON(3)   0            35  5 (Digit)
307 1E77F 0000        REL(5) =ONEDGT
          0
308 1E784 0           NIBHEX 0
309           *
310           =a6     EQU    #36
311 1E785 000         CON(3)   0            36  6 (Digit)
312 1E788 0000        REL(5) =ONEDGT
          0
313 1E78D 0           NIBHEX 0
314           *
315           =a7     EQU    #37
316 1E78E 000         CON(3)   0            37  7 (Digit)
317 1E791 0000        REL(5) =ONEDGT
          0
318 1E796 0           NIBHEX 0
319           *
320           =a8     EQU    #38
321 1E797 000         CON(3)   0            38  8 (Digit)
322 1E79A 0000        REL(5) =ONEDGT
          0
323 1E79F 0           NIBHEX 0
```

```
324                *
325                =a9     EQU    #39
326 1E7A0 000              CON(3)    0          39  9 (Digit)
327 1E7A3 0000             REL(5) =ONEDGT
          0
328 1E7A8 0                NIBHEX 0
329                *
330 1E7A9 000              CON(3)    0          3A  (:)
331 1E7AC 0000             REL(5) =TRMNTR
          0
332 1E7B1 0                NIBHEX 0
333                *
334 1E7B2 000              CON(3)    0          3B  (;)
335 1E7B5 0000             REL(5) =TRMNTR
          0
336 1E7BA 0                NIBHEX 0
337                *
338 1E7BB 000              CON(3)    0          3C  (<)
339 1E7BE 0000             REL(5) =TRMNTR
          0
340 1E7C3 0                NIBHEX 0
341                *
342 1E7C4 000              CON(3)    0          3D  CALC MODE ASNMNT OPRTR  (=)
343 1E7C7 0000             REL(5) =TRMNTR
          0
344 1E7CC 0                NIBHEX 0
345                *
346 1E7CD 000              CON(3)    0          3E  (>)
347 1E7D0 0000             REL(5) =TRMNTR
          0
348 1E7D5 0                NIBHEX 0
349                *
350 1E7D6 000              CON(3)    0          3F  (?)
351 1E7D9 0000             REL(5) =TRMNTR
          0
352 1E7DE 0                NIBHEX 0
353                *
354 1E7DF 000              CON(3)    0          40  (@)
355 1E7E2 0000             REL(5) =TRMNTR
          0
356 1E7E7 0                NIBHEX 0
357                *
358 1E7E8 000              CON(3)    0          41  A (Static Variable)
359 1E7EB 0000             REL(5) =STATIC
          0
360 1E7F0 0                NIBHEX 0
361                *
362 1E7F1 000              CON(3)    0          42  B (Static Variable)
363 1E7F4 0000             REL(5) =STATIC
          0
364 1E7F9 0                NIBHEX 0
365                *
366 1E7FA 000              CON(3)    0          43  C (Static Variable)
367 1E7FD 0000             REL(5) =STATIC
          0
```

```
368 1E802 0           NIBHEX 0
369           *
370 1E803 000         CON(3)   0          44  D (Static Variable)
371 1E806 0000        REL(5) =STATIC
          0
372 1E80B 0           NIBHEX 0
373           *
374 1E80C 000         CON(3)   0          45  E (Static Variable)
375 1E80F 0000        REL(5) =STATIC
          0
376 1E814 0           NIBHEX 0
377           *
378 1E815 000         CON(3)   0          46  F (Static Variable)
379 1E818 0000        REL(5) =STATIC
          0
380 1E81D 0           NIBHEX 0
381           *
382 1E81E 000         CON(3)   0          47  G (Static Variable)
383 1E821 0000        REL(5) =STATIC
          0
384 1E826 0           NIBHEX 0
385           *
386 1E827 000         CON(3)   0          48  H (Static Variable)
387 1E82A 0000        REL(5) =STATIC
          0
388 1E82F 0           NIBHEX 0
389           *
390 1E830 000         CON(3)   0          49  I (Static Variable)
391 1E833 0000        REL(5) =STATIC
          0
392 1E838 0           NIBHEX 0
393           *
394 1E839 000         CON(3)   0          4A  J (Static Variable)
395 1E83C 0000        REL(5) =STATIC
          0
396 1E841 0           NIBHEX 0
397           *
398 1E842 000         CON(3)   0          4B  K (Static Variable)
399 1E845 0000        REL(5) =STATIC
          0
400 1E84A 0           NIBHEX 0
401           *
402 1E84B 000         CON(3)   0          4C  L (Static Variable)
403 1E84E 0000        REL(5) =STATIC
          0
404 1E853 0           NIBHEX 0
405           *
406 1E854 000         CON(3)   0          4D  M (Static Variable)
407 1E857 0000        REL(5) =STATIC
          0
408 1E85C 0           NIBHEX 0
409           *
410 1E85D 000         CON(3)   0          4E  N (Static Variable)
411 1E860 0000        REL(5) =STATIC
          0
```

```
412 1E865 0           NIBHEX 0
413           *
414 1E866 000         CON(3)   0          4F  O (Static Variable)
415 1E869 0000        REL(5) =STATIC
          0
416 1E86E 0           NIBHEX 0
417           *
418 1E86F 000         CON(3)   0          50  P (Static Variable)
419 1E872 0000        REL(5) =STATIC
          0
420 1E877 0           NIBHEX 0
421           *
422 1E878 000         CON(3)   0          51  Q (Static Variable)
423 1E87B 0000        REL(5) =STATIC
          0
424 1E880 0           NIBHEX 0
425           *
426 1E881 000         CON(3)   0          52  R (Static Variable)
427 1E884 0000        REL(5) =STATIC
          0
428 1E889 0           NIBHEX 0
429           *
430 1E88A 000         CON(3)   0          53  S (Static Variable)
431 1E88D 0000        REL(5) =STATIC
          0
432 1E892 0           NIBHEX 0
433           *
434 1E893 000         CON(3)   0          54  T (Static Variable)
435 1E896 0000        REL(5) =STATIC
          0
436 1E89B 0           NIBHEX 0
437           *
438 1E89C 000         CON(3)   0          55  U (Static Variable)
439 1E89F 0000        REL(5) =STATIC
          0
440 1E8A4 0           NIBHEX 0
441           *
442 1E8A5 000         CON(3)   0          56  V (Static Variable)
443 1E8A8 0000        REL(5) =STATIC
          0
444 1E8AD 0           NIBHEX 0
445           *
446 1E8AE 000         CON(3)   0          57  W (Static Variable)
447 1E8B1 0000        REL(5) =STATIC
          0
448 1E8B6 0           NIBHEX 0
449           *
450 1E8B7 000         CON(3)   0          58  X (Static Variable)
451 1E8BA 0000        REL(5) =STATIC
          0
452 1E8BF 0           NIBHEX 0
453           *
454 1E8C0 000         CON(3)   0          59  Y (Static Variable)
455 1E8C3 0000        REL(5) =STATIC
          0
```

```
456 1E8C8 0             NIBHEX 0
457               *
458               =tZ     EQU    #5A
459 1E8C9 000           CON(3)   0            5A  Z (Static Variable)
460 1E8CC 0000          REL(5) =STATIC
          0
461 1E8D1 0             NIBHEX 0
462               *
463 1E8D2 000           CON(3)   0            5B  ([)
464 1E8D5 0000          REL(5) =TRMNTR
          0
465 1E8DA 0             NIBHEX 0
466               *
467 1E8DB 000           CON(3)   0            5C  (\)
468 1E8DE 0000          REL(5) =TRMNTR
          0
469 1E8E3 0             NIBHEX 0
470               *
471 1E8E4 000           CON(3)   0            5D  (])
472 1E8E7 0000          REL(5) =TRMNTR
          0
473 1E8EC 0             NIBHEX 0
474               *
475 1E8ED 000           CON(3)   0            5E  (^)
476 1E8F0 0000          REL(5) =TRMNTR
          0
477 1E8F5 0             NIBHEX 0
478               *
479 1E8F6 000           CON(3)   0            5F  (_)
480 1E8F9 0000          REL(5) =TRMNTR
          0
481 1E8FE 0             NIBHEX 0
482               *
483               =tADIG0 EQU    #60
484 1E8FF 000           CON(3)   0            60  Dynamic Variable 0
485 1E902 0000          REL(5) =DYNAMC
          0
486 1E907 0             NIBHEX 0
487               *
488               =tADIG1 EQU    #61
489 1E908 000           CON(3)   0            61  Dynamic Variable 1
490 1E90B 0000          REL(5) =DYNAMC
          0
491 1E910 0             NIBHEX 0
492               *
493               =tADIG2 EQU    #62
494 1E911 000           CON(3)   0            62  Dynamic Variable 2
495 1E914 0000          REL(5) =DYNAMC
          0
496 1E919 0             NIBHEX 0
497               *
498               =tADIG3 EQU    #63
499 1E91A 000           CON(3)   0            63  Dynamic Variable 3
500 1E91D 0000          REL(5) =DYNAMC
          0
```

```
501 1E922 0           NIBHEX 0
502           *
503           =tADIG4 EQU    #64
504 1E923 000         CON(3)    0          64  Dynamic Variable 4
505 1E926 0000        REL(5) =DYNAMC
          0
506 1E92B 0           NIBHEX 0
507           *
508           =tADIG5 EQU    #65
509 1E92C 000         CON(3)    0          65  Dynamic Variable 5
510 1E92F 0000        REL(5) =DYNAMC
          0
511 1E934 0           NIBHEX 0
512           *
513           =tADIG6 EQU    #66
514 1E935 000         CON(3)    0          66  Dynamic Variable 6
515 1E938 0000        REL(5) =DYNAMC
          0
516 1E93D 0           NIBHEX 0
517           *
518           =tADIG7 EQU    #67
519 1E93E 000         CON(3)    0          67  Dynamic Variable 7
520 1E941 0000        REL(5) =DYNAMC
          0
521 1E946 0           NIBHEX 0
522           *
523           =tADIG8 EQU    #68
524 1E947 000         CON(3)    0          68  Dynamic Variable 8
525 1E94A 0000        REL(5) =DYNAMC
          0
526 1E94F 0           NIBHEX 0
527           *
528           =tADIG9 EQU    #69
529 1E950 000         CON(3)    0          69  Dynamic Variable 9
530 1E953 0000        REL(5) =DYNAMC
          0
531 1E958 0           NIBHEX 0
532           *
533           =tIP    EQU    #6A
534 1E959 F42         CON(3)  591          6A  IP
535 1E95C 0000        REL(5) =IP
          0
536 1E961 F           NIBHEX F
537           *
538           =tFP    EQU    #6B
539 1E962 CD1         CON(3)  476          6B  FP
540 1E965 0000        REL(5) =FP
          0
541 1E96A F           NIBHEX F
542           *
543           =tMAXRL EQU    #6C
544 1E96B 0E2         CON(3)  736          6C  MAXREAL
545 1E96E 0000        REL(5) =MAXRL
          0
546 1E973 F           NIBHEX F
```

```
547                *
548                =tRMD   EQU    #6D
549 1E974 614              CON(3) 1046            6D  RMD
550 1E977 0000             REL(5) =RMD
          0
551 1E97C F                NIBHEX F
552                *
553                =tRAD   EQU    #6E
554 1E97D 5C3              CON(3)  965            6E  RAD
555 1E980 0000             REL(5) =RAD
          0
556 1E985 F                NIBHEX F
557                *
558                =tDEG   EQU    #6F
559 1E986 9F0              CON(3)  249            6F  DEG
560 1E989 0000             REL(5) =DEG
          0
561 1E98E F                NIBHEX F
562                *
563                =tINF   EQU    #70
564 1E98F 612              CON(3)  534            70  INF
565 1E992 0000             REL(5) =INF
          0
566 1E997 F                NIBHEX F
567                *
568                =tEPS   EQU    #71
569 1E998 471              CON(3)  372            71  EPS
570 1E99B 0000             REL(5) =EPS
          0
571 1E9A0 F                NIBHEX F
572                *
573                =tCEIL  EQU    #72
574 1E9A1 580              CON(3)  133            72  CEIL
575 1E9A4 0000             REL(5) =CEIL
          0
576 1E9A9 F                NIBHEX F
577                *
578                =tKEY$  EQU    #73
579 1E9AA 662              CON(3)  614            73  KEY$
580 1E9AD 0000             REL(5) =KEY$
          0
581 1E9B2 F                NIBHEX F
582                *
583                =tMOD   EQU    #74
584 1E9B3 E03              CON(3)  782            74  MOD
585 1E9B6 0000             REL(5) =MOD
          0
586 1E9BB F                NIBHEX F
587                *
588                =tERRL  EQU    #75
589 1E9BC D71              CON(3)  381            75  ERRL
590 1E9BF 0000             REL(5) =ERRL
          0
591 1E9C4 F                NIBHEX F
592                *
```

```
593                =tERRN  EQU    #76
594 1E9C5 881              CON(3)  392            76  ERRN
595 1E9C8 0000             REL(5) =ERRN
          0
596 1E9CD F                NIBHEX F
597                *
598                =tDATE  EQU    #77
599 1E9CE 4D0              CON(3)  212            77  DATE
600 1E9D1 0000             REL(5) =DATE
          0
601 1E9D6 F                NIBHEX F
602                *
603                =tDATE$ EQU    #78
604 1E9D7 7C0              CON(3)  199            78  DATE$
605 1E9DA 0000             REL(5) =DATE$
          0
606 1E9DF F                NIBHEX F
607                *
608                =tPI    EQU    #79
609 1E9E0 B73              CON(3)  891            79  PI
610 1E9E3 0000             REL(5) =PI
          0
611 1E9E8 F                NIBHEX F
612                *
613                =tCMPLX EQU    #7A
614 1E9E9 000              CON(3)    0            7A  CMPLX
615 1E9EC 0000             REL(5) =CMPLX
          0
616 1E9F1 F                NIBHEX F
617                *
618                =tTIME  EQU    #7B
619 1E9F2 DD4              CON(3) 1245            7B  TIME
620 1E9F5 0000             REL(5) =TIME
          0
621 1E9FA F                NIBHEX F
622                *
623                =tFN    EQU    #7C
624 1E9FB 000              CON(3)    0            7C  FN
625 1E9FE 0000             REL(5) =FN
          0
626 1EA03 0                NIBHEX 0
627                *
628                =tARRAY EQU    #7D
629 1EA04 000              CON(3)    0            7D  ARRAY
630 1EA07 0000             REL(5) =ARRAY
          0
631 1EA0C 0                NIBHEX 0
632                *
633                =tDMYAR EQU    #7E
634 1EA0D 000              CON(3)    0            7E  Dummy array
635 1EA10 0000             REL(5) =DMARRY
          0
636 1EA15 0                NIBHEX 0
637                *
638                =tRES   EQU    #7F
```

```
    639 1EA16 EF3          CON(3) 1022           7F  RES
    640 1EA19 0000         REL(5) =RES
          0
    641 1EA1E F            NIBHEX F
    642             *
    643             =t^    EQU    #80
    644 1EA1F 000          CON(3)   0            80  ^ (INVOLUTION)
    645 1EA22 0000         REL(5) =INVLUT
          0
    646 1EA27 0            NIBHEX 0
    647             *
    648             =tNOT  EQU    #81
    649 1EA28 D23          CON(3)  813           81  NOT
    650 1EA2B 0000         REL(5) =NOT
          0
    651 1EA30 0            NIBHEX 0
    652             *
    653             =t-    EQU    #82
    654 1EA31 000          CON(3)   0            82  - (Unary)
    655 1EA34 0000         REL(5) =MINUS
          0
    656 1EA39 0            NIBHEX 0
    657             *
    658             =t*    EQU    #83
    659 1EA3A 000          CON(3)   0            83  *
    660 1EA3D 0000         REL(5) =MULTPY
          0
    661 1EA42 0            NIBHEX 0
    662             *
    663             =t/    EQU    #84
    664 1EA43 000          CON(3)   0            84  /
    665 1EA46 0000         REL(5) =DIVIDE
          0
    666 1EA4B 0            NIBHEX 0
    667             *
    668             =t%    EQU    #85
    669 1EA4C 000          CON(3)   0            85  %
    670 1EA4F 0000         REL(5) =PERCNT
          0
    671 1EA54 0            NIBHEX 0
    672             *
    673             =tDIV  EQU    #86
    674 1EA55 341          CON(3)  323           86  DIV
    675 1EA58 0000         REL(5) =DIV
          0
    676 1EA5D 0            NIBHEX 0
    677             *
    678             =t+    EQU    #87
    679 1EA5E 000          CON(3)   0            87  +
    680 1EA61 0000         REL(5) =PLUS
          0
    681 1EA66 0            NIBHEX 0
    682             *
    683 1EA67 000          CON(3)   0            88  [Unused]
    684 1EA6A 0000         NIBHEX 00000
```

```
          0
685 1EA6F 0           NIBHEX 0
686             *
687             =t&     EQU    #89
688 1EA70 000         CON(3)    0          89  & (CONCATENATE)
689 1EA73 0000        REL(5) =CONCAT
          0
690 1EA78 0           NIBHEX 0
691             *
692             =tRELOP EQU    #8A
693 1EA79 000         CON(3)    0          8A  Relational operators
694 1EA7C 0000        REL(5) =COMPAR
          0
695 1EA81 0           NIBHEX 0
696             *
697             =tAND   EQU    #8B
698 1EA82 620         CON(3)   38          8B  AND
699 1EA85 0000        REL(5) =AND
          0
700 1EA8A 0           NIBHEX 0
701             *
702             =tEXOR  EQU    #8C
703 1EA8B 0A1         CON(3)  416          8C  EXOR
704 1EA8E 0000        REL(5) =EXOR
          0
705 1EA93 0           NIBHEX 0
706             *
707             =tOR    EQU    #8D
708 1EA94 E53         CON(3)  862          8D  OR
709 1EA97 0000        REL(5) =OR
          0
710 1EA9C 0           NIBHEX 0
711             *
712 1EA9D 000         CON(3)    0          8E  [Unused]
713 1EAA0 0000        REL(5) =EXPR
          0
714 1EAA5 0           NIBHEX 0
715             *
716 1EAA6 000         CON(3)    0          8F  [Unused]
717 1EAA9 0000        REL(5) =EXPR
          0
718 1EAAE 0           NIBHEX 0
719             *
720             =tLOG   EQU    #90
721 1EAAF 5C2         CON(3)  709          90  LOG
722 1EAB2 0000        REL(5) =LOG
          0
723 1EAB7 F           NIBHEX F
724             *
725             =tLN    EQU    #91
726 1EAB8 1B2         CON(3)  689          91  LN
727 1EABB 0000        REL(5) =LOG
          0
728 1EAC0 F           NIBHEX F
729             *
```

```
730             =tSQR   EQU    #92
731 1EAC1 864           CON(3) 1128            92  SQR
732 1EAC4 0000          REL(5) =SQR
          0
733 1EAC9 F             NIBHEX F
734             *
735             =tLOG10 EQU    #93
736 1EACA 8B2           CON(3)  696            93  LOG10
737 1EACD 0000          REL(5) =LOG10
          0
738 1EAD2 F             NIBHEX F
739             *
740             =tEXP   EQU    #94
741 1EAD3 BA1           CON(3)  427            94  EXP
742 1EAD6 0000          REL(5) =EXP
          0
743 1EADB F             NIBHEX F
744             *
745             =tTIME$ EQU    #95
746 1EADC 3C4           CON(3) 1219            95  TIME$
747 1EADF 0000          REL(5) =TIME$
          0
748 1EAE4 F             NIBHEX F
749             *
750             =tSIN   EQU    #96
751 1EAE5 F54           CON(3) 1119            96  SIN
752 1EAE8 0000          REL(5) =SIN
          0
753 1EAED F             NIBHEX F
754             *
755             =tCOS   EQU    #97
756 1EAEE 3B0           CON(3)  179            97  COS
757 1EAF1 0000          REL(5) =COS
          0
758 1EAF6 F             NIBHEX F
759             *
760             =tTAN   EQU    #98
761 1EAF7 FA4           CON(3) 1199            98  TAN
762 1EAFA 0000          REL(5) =TAN
          0
763 1EAFF F             NIBHEX F
764             *
765             =tASIN  EQU    #99
766 1EB00 F20           CON(3)   47            99  ASIN
767 1EB03 0000          REL(5) =ASIN
          0
768 1EB08 F             NIBHEX F
769             *
770             =tACOS  EQU    #9A
771 1EB09 900           CON(3)    9            9A  ACOS
772 1EB0C 0000          REL(5) =ACOS
          0
773 1EB11 F             NIBHEX F
774             *
775             =tATAN  EQU    #9B
```

```
776 1EB12 A30          CON(3)   58          9B  ATAN
777 1EB15 0000         REL(5) =ATAN
          0
778 1EB1A F            NIBHEX F
779             *
780             =tINT  EQU    #9C
781 1EB1B D32          CON(3)  573          9C  INT
782 1EB1E 0000         REL(5) =INT
          0
783 1EB23 F            NIBHEX F
784             *
785             =tMEAN EQU    #9D
786 1EB24 AF2          CON(3)  762          9D  MEAN
787 1EB27 0000         REL(5) =MEAN
          0
788 1EB2C F            NIBHEX F
789             *
790             =tSDEV EQU    #9E
791 1EB2D 134          CON(3) 1073          9E  SDEV
792 1EB30 0000         REL(5) =SDEV
          0
793 1EB35 F            NIBHEX F
794             *
795             =tPREDV EQU   #9F
796 1EB36 D83          CON(3)  909          9F  PREDV
797 1EB39 0000         REL(5) =PREDV
          0
798 1EB3E F            NIBHEX F
799             *
800             =tRND  EQU    #A0
801 1EB3F F14          CON(3) 1055          A0  RND
802 1EB42 0000         REL(5) =RND
          0
803 1EB47 F            NIBHEX F
804             *
805             =tSGN  EQU    #A1
806 1EB48 944          CON(3) 1097          A1  SGN
807 1EB4B 0000         REL(5) =SGN
          0
808 1EB50 F            NIBHEX F
809             *
810             =tABS  EQU    #A2
811 1EB51 000          CON(3)    0          A2  ABS
812 1EB54 0000         REL(5) =ABS
          0
813 1EB59 F            NIBHEX F
814             *
815             =tNUM  EQU    #A3
816 1EB5A 633          CON(3)  822          A3  NUM
817 1EB5D 0000         REL(5) =NUM
          0
818 1EB62 F            NIBHEX F
819             *
820             =tCHR$ EQU    #A4
821 1EB63 D90          CON(3)  157          A4  CHR$
```

```
    822 1EB66 0000          REL(5) =CHR$
              0
    823 1EB6B F             NIBHEX F
    824               *
    825               =tVAL EQU    #A5
    826 1EB6C A25           CON(3) 1322           A5  VAL
    827 1EB6F 0000          REL(5) =VAL
              0
    828 1EB74 F             NIBHEX F
    829               *
    830               =tSTR$ EQU   #A6
    831 1EB75 294           CON(3) 1170           A6  STR$ (formerly VAL$)
    832 1EB78 0000          REL(5) =STR$
              0
    833 1EB7D F             NIBHEX F
    834               *
    835               =tISUB$ EQU  #A7
    836 1EB7E 000           CON(3)   0            A7  SUB$ (implied)
    837 1EB81 0000          REL(5) =SUB$
              0
    838 1EB86 F             NIBHEX F
    839               *
    840               =tFACT EQU   #A8
    841 1EB87 4B1           CON(3)  436           A8  FACT
    842 1EB8A 0000          REL(5) =FACT
              0
    843 1EB8F F             NIBHEX F
    844               *
    845               =tLEN  EQU   #A9
    846 1EB90 582           CON(3)  645           A9  LEN
    847 1EB93 0000          REL(5) =LEN
              0
    848 1EB98 F             NIBHEX F
    849               *
    850               =tLPRP EQU   #AA
    851 1EB99 000           CON(3)   0            AA  LPRP  ()
    852 1EB9C 0000          REL(5) =LPRP
              0
    853 1EBA1 F             NIBHEX F
    854               *
    855               =tUPRC$ EQU  #AB
    856 1EBA2 505           CON(3) 1285           AB  UPRC$
    857 1EBA5 0000          REL(5) =UPRC$
              0
    858 1EBAA F             NIBHEX F
    859               *
    860               =tMIN  EQU   #AC
    861 1EBAB 503           CON(3)  773           AC  MIN
    862 1EBAE 0000          REL(5) =MIN
              0
    863 1EBB3 F             NIBHEX F
    864               *
    865               =tMAX  EQU   #AD
    866 1EBB4 1F2           CON(3)  753           AD  MAX
    867 1EBB7 0000          REL(5) =MAX
```

```
          0
  868 1EBBC F             NIBHEX F
  869             *
  870             =tIVL   EQU    #AE
  871 1EBBD D52           CON(3)  605            AE  IVL
  872 1EBC0 0000          REL(5) =IVL
          0
  873 1EBC5 F             NIBHEX F
  874             *
  875             =tOVF   EQU    #AF
  876 1EBC6 563           CON(3)  869            AF  OVF
  877 1EBC9 0000          REL(5) =OVF
          0
  878 1EBCE F             NIBHEX F
  879             *
  880             =tUNF   EQU    #B0
  881 1EBCF CF4           CON(3) 1276            B0  UNF
  882 1EBD2 0000          REL(5) =UNF
          0
  883 1EBD7 F             NIBHEX F
  884             *
  885             =tDVZ   EQU    #B1
  886 1EBD8 C41           CON(3)  332            B1  DVZ
  887 1EBDB 0000          REL(5) =DVZ
          0
  888 1EBE0 F             NIBHEX F
  889             *
  890             =tINX   EQU    #B2
  891 1EBE1 642           CON(3)  582            B2  INX
  892 1EBE4 0000          REL(5) =INX
          0
  893 1EBE9 F             NIBHEX F
  894             *
  895             =tXFN   EQU    #B3
  896 1EBEA 000           CON(3)    0            B3  XFN
  897 1EBED 0000          REL(5) =XFN
          0
  898 1EBF2 F             NIBHEX F
  899             *
  900             =tFFN   EQU    #B4
  901 1EBF3 000           CON(3)    0            B4  Funny Function
  902 1EBF6 0000          REL(5) =XFN
          0
  903 1EBFB F             NIBHEX F
  904             *
  905             =tCOPY  EQU    #B5
  906 1EBFC 8A0           CON(3)  168            B5  COPY
  907 1EBFF 0000          REL(5) =COPY
          0
  908 1EC04 D             NIBHEX D
  909             *
  910             =tLR    EQU    #B6
  911 1EC05 EC2           CON(3)  718            B6  LR
  912 1EC08 0000          REL(5) =LR
          0
```

```
913 1EC0D D           NIBHEX D
914           *
915           =tDELET EQU    #B7
916 1EC0E F01         CON(3)  271          B7  DELETE
917 1EC11 0000        REL(5) =D'LTE
          0
918 1EC16 7           NIBHEX 7
919           *
920           =tEDIT  EQU    #B8
921 1EC17 551         CON(3)  341          B8  EDIT
922 1EC1A 0000        REL(5) =EDIT
          0
923 1EC1F 7           NIBHEX 7
924           *
925           =tDEF   EQU    #B9
926 1EC20 FD0         CON(3)  223          B9  DEF
927 1EC23 0000        REL(5) =DEF
          0
928 1EC28 D           NIBHEX D
929           *
930           =tENDDF EQU    #BA
931 1EC29 000         CON(3)   0           BA  END DEF (parsed by ENDP)
932 1EC2C 0000        REL(5) =ENDDEF
          0
933 1EC31 0           NIBHEX 0
934           *
935           =tLIST  EQU    #BB
936 1EC32 6A2         CON(3)  678          BB  LIST
937 1EC35 0000        REL(5) =LIST
          0
938 1EC3A D           NIBHEX D
939           *
940           =tREAL  EQU    #BC
941 1EC3B 9D3         CON(3)  985          BC  REAL
942 1EC3E 0000        REL(5) =REAL
          0
943 1EC43 D           NIBHEX D
944           *
945           =tNAME  EQU    #BD
946 1EC44 713         CON(3)  791          BD  NAME
947 1EC47 0000        REL(5) =NAME
          0
948 1EC4C D           NIBHEX D
949           *
950           =tDSTRY EQU    #BE
951 1EC4D E11         CON(3)  286          BE  DESTROY
952 1EC50 0000        REL(5) =DSTROY
          0
953 1EC55 D           NIBHEX D
954           *
955           =tLINPT EQU    #BF
956 1EC56 792         CON(3)  663          BF  LINPUT
957 1EC59 0000        REL(5) =LINPUT
          0
958 1EC5E D           NIBHEX D
```

```
 959                *
 960                =tLET   EQU    #C0
 961 1EC5F E82              CON(3)  654          C0  LET
 962 1EC62 0000             REL(5) =LET
           0
 963 1EC67 D                NIBHEX D
 964                *
 965                =tSUB   EQU    #C1
 966 1EC68 D94              CON(3) 1181          C1  SUB
 967 1EC6B 0000             REL(5) =SUB
           0
 968 1EC70 8                NIBHEX 8
 969                *
 970                =tENDSB EQU    #C2
 971 1EC71 000              CON(3)   0           C2  END SUB (parsed by ENDP)
 972 1EC74 0000             REL(5) =ENDSUB
           0
 973 1EC79 0                NIBHEX 0
 974                *
 975                =tFOR   EQU    #C3
 976 1EC7A 3D1              CON(3)  467          C3  FOR
 977 1EC7D 0000             REL(5) =FOR
           0
 978 1EC82 9                NIBHEX 9
 979                *
 980                =tNEXT  EQU    #C4
 981 1EC83 223              CON(3)  802          C4  NEXT
 982 1EC86 0000             REL(5) =NEXT
           0
 983 1EC8B 9                NIBHEX 9
 984                *
 985                =tDISP  EQU    #C5
 986 1EC8C 831              CON(3)  312          C5  DISP
 987 1EC8F 0000             REL(5) =DISP
           0
 988 1EC94 D                NIBHEX D
 989                *
 990                =tDATA  EQU    #C6
 991 1EC95 CB0              CON(3)  188          C6  DATA
 992 1EC98 0000             REL(5) =DATA
           0
 993 1EC9D 8                NIBHEX 8
 994                *
 995                =tREAD  EQU    #C7
 996 1EC9E EC3              CON(3)  974          C7  READ
 997 1ECA1 0000             REL(5) =READ
           0
 998 1ECA6 D                NIBHEX D
 999                *
1000                =tFETCH EQU    #C8
1001 1ECA7 FB1              CON(3)  447          C8  FETCH
1002 1ECAA 0000             REL(5) =FETCH
           0
1003 1ECAF 7                NIBHEX 7
1004                *
```

```
1005            =tINPUT EQU    #C9
1006 1ECB0 F12          CON(3)  543            C9  INPUT
1007 1ECB3 0000         REL(5) =INPUT
          0
1008 1ECB8 D            NIBHEX D
1009            *
1010            =tINTEG EQU    #CA
1011 1ECB9 C22          CON(3)  556            CA  INTEGER
1012 1ECBC 0000         REL(5) =INTEGR
          0
1013 1ECC1 D            NIBHEX D
1014            *
1015            =tSHORT EQU    #CB
1016 1ECC2 254          CON(3) 1106            CB  SHORT
1017 1ECC5 0000         REL(5) =SHORT
          0
1018 1ECCA D            NIBHEX D
1019            *
1020            =tDIM   EQU    #CC
1021 1ECCB F21          CON(3)  303            CC  DIM
1022 1ECCE 0000         REL(5) =DIM
          0
1023 1ECD3 D            NIBHEX D
1024            *
1025            =tPRINT EQU    #CD
1026 1ECD4 A93          CON(3)  922            CD  PRINT
1027 1ECD7 0000         REL(5) =PRINT
          0
1028 1ECDC D            NIBHEX D
1029            *
1030            =tSTAT  EQU    #CE
1031 1ECDD 174          CON(3) 1137            CE  STAT
1032 1ECE0 0000         REL(5) =STAT
          0
1033 1ECE5 D            NIBHEX D
1034            *
1035            =tKEYS  EQU    #CF
1036 1ECE6 172          CON(3)  625            CF  KEYS
1037 1ECE9 0000         NIBHEX 00000
          0
1038 1ECEE 0            NIBHEX 0
1039            *
1040            =tCARD  EQU    #D0
1041 1ECEF 170          CON(3)  113            D0  CARD
1042 1ECF2 0000         NIBHEX 00000
          0
1043 1ECF7 0            NIBHEX 0
1044            *
1045            =tPORT  EQU    #D1
1046 1ECF8 283          CON(3)  898            D1  PORT
1047 1ECFB 0000         NIBHEX 00000
          0
1048 1ED00 0            NIBHEX 0
1049            *
1050            =tMAIN  EQU    #D2
```

```
1051 1ED01 5D2          CON(3)  725          D2  MAIN
1052 1ED04 0000         NIBHEX 00000
           0
1053 1ED09 0            NIBHEX 0
1054               *
1055               =tDEGRE EQU    #D3
1056 1ED0A 8E0          CON(3)  232          D3  DEGREES
1057 1ED0D 0000         REL(5) =DEGREE
           0
1058 1ED12 D            NIBHEX D
1059               *
1060               =tRDIAN EQU    #D4
1061 1ED13 4B3          CON(3)  948          D4  RADIANS
1062 1ED16 0000         REL(5) =RADIAN
           0
1063 1ED1B D            NIBHEX D
1064               *
1065               =tADD   EQU    #D5
1066 1ED1C 410          CON(3)   20          D5  ADD
1067 1ED1F 0000         REL(5) =ADD
           0
1068 1ED24 D            NIBHEX D
1069               *
1070               =tDELAY EQU    #D6
1071 1ED25 201          CON(3)  258          D6  DELAY
1072 1ED28 0000         REL(5) =DELAY
           0
1073 1ED2D D            NIBHEX D
1074               *
1075               =tPAUSE EQU    #D7
1076 1ED2E E63          CON(3)  878          D7  PAUSE
1077 1ED31 0000         REL(5) =PAUSE
           0
1078 1ED36 C            NIBHEX C
1079               *
1080               =tWAIT  EQU    #D8
1081 1ED37 335          CON(3) 1331          D8  WAIT
1082 1ED3A 0000         REL(5) =WAIT
           0
1083 1ED3F D            NIBHEX D
1084               *
1085               =tSTOP  EQU    #D9
1086 1ED40 784          CON(3) 1159          D9  STOP
1087 1ED43 0000         REL(5) =STOP
           0
1088 1ED48 D            NIBHEX D
1089               *
1090               =tEND   EQU    #DA
1091 1ED49 B61          CON(3)  363          DA  END
1092 1ED4C 0000         REL(5) =END
           0
1093 1ED51 D            NIBHEX D
1094               *
1095               =tRETRN EQU    #DB
1096 1ED52 704          CON(3) 1031          DB  RETURN
```

```
1097 1ED55 0000          REL(5) =RETURN
           0
1098 1ED5A D             NIBHEX D
1099               *
1100               =tGOSUB EQU    #DC
1101 1ED5B 3E1           CON(3)  483           DC  GOSUB
1102 1ED5E 0000          REL(5) =GOSUB
           0
1103 1ED63 D             NIBHEX D
1104               *
1105               =tGOTO  EQU    #DD
1106 1ED64 0F1           CON(3)  496           DD  GOTO
1107 1ED67 0000          REL(5) =GOTO
           0
1108 1ED6C D             NIBHEX D
1109               *
1110               =tRESTR EQU    #DE
1111 1ED6D DE3           CON(3) 1005           DE  RESTORE
1112 1ED70 0000          REL(5) =RESTOR
           0
1113 1ED75 D             NIBHEX D
1114               *
1115               =tIF    EQU    #DF
1116 1ED76 202           CON(3)  514           DF  IF
1117 1ED79 0000          REL(5) =IF
           0
1118 1ED7E D             NIBHEX D
1119               *
1120               =tON    EQU    #E0
1121 1ED7F 843           CON(3)  840           E0  ON
1122 1ED82 0000          REL(5) =ON
           0
1123 1ED87 D             NIBHEX D
1124               *
1125               =tOFF   EQU    #E1
1126 1ED88 F33           CON(3)  831           E1  OFF
1127 1ED8B 0000          REL(5) =OFF
           0
1128 1ED90 D             NIBHEX D
1129               *
1130               =tUSER  EQU    #E2
1131 1ED91 215           CON(3) 1298           E2  USER
1132 1ED94 0000          REL(5) =USER
           0
1133 1ED99 D             NIBHEX D
1134               *
1135               =tERROR EQU    #E3
1136 1ED9A 391           CON(3)  403           E3  ERROR
1137 1ED9D 0000          REL(5) =NXTSTM
           0
1138 1EDA2 0             NIBHEX 0
1139               *
1140               =tTIMER EQU    #E4
1141 1EDA3 0D4           CON(3) 1232           E4  TIMER
1142 1EDA6 0000          REL(5) =NXTSTM
```

```
           0
1143 1EDAB 0            NIBHEX 0
1144               *
1145               =tKEY   EQU    #E5
1146 1EDAC C72          CON(3)  636            E5  KEY
1147 1EDAF 0000         REL(5) =KEY
           0
1148 1EDB4 D            NIBHEX D
1149               *
1150               =tREM   EQU    #E6
1151 1EDB5 4E3          CON(3)  996            E6  REM
1152 1EDB8 0000         REL(5) =REM
           0
1153 1EDBD D            NIBHEX D
1154               *
1155               =tIS    EQU    #E7
1156 1EDBE 652          CON(3)  598            E7  IS
1157 1EDC1 0000         REL(5) =NXTSTM
           0
1158 1EDC6 0            NIBHEX 0
1159               *
1160               =tBEEP  EQU    #E8
1161 1EDC7 B50          CON(3)   91            E8  BEEP
1162 1EDCA 0000         REL(5) =BEEP
           0
1163 1EDCF D            NIBHEX D
1164               *
1165               =tBASE  EQU    #E9
1166 1EDD0 050          CON(3)   80            E9  BASE
1167 1EDD3 0000         REL(5) =NXTSTM
           0
1168 1EDD8 0            NIBHEX 0
1169               *
1170               =tTRACE EQU    #EA
1171 1EDD9 FE4          CON(3) 1263            EA  TRACE
1172 1EDDC 0000         REL(5) =TRACE
           0
1173 1EDE1 D            NIBHEX D
1174               *
1175               =tPURGE EQU    #EB
1176 1EDE2 7A3          CON(3)  935            EB  PURGE
1177 1EDE5 0000         REL(5) =PURGE
           0
1178 1EDEA D            NIBHEX D
1179               *
1180               =tCAT   EQU    #EC
1181 1EDEB C70          CON(3)  124            EC  CAT
1182 1EDEE 0000         REL(5) =CAT
           0
1183 1EDF3 D            NIBHEX D
1184               *
1185               =tOPT'N EQU    #ED
1186 1EDF4 F43          CON(3)  847            ED  OPTION
1187 1EDF7 0000         REL(5) =OPTION
           0
```

```
1188 1EDFC D            NIBHEX D
1189            *
1190            =tAUTO  EQU    #EE
1191 1EDFD 540          CON(3)   69            EE  AUTO
1192 1EE00 0000         REL(5) =AUTO
           0
1193 1EE05 7            NIBHEX 7
1194            *
1195            =tXWORD EQU    #EF
1196 1EE06 000          CON(3)    0            EF  XWORD
1197 1EE09 0000         REL(5) =XWORD
           0
1198 1EE0E D            NIBHEX D
1199            *
1200            =tEOL   EQU    #F0
1201 1EE0F 000          CON(3)    0            F0  <eol>
1202 1EE12 0000         REL(5) =TRMNTR
           0
1203 1EE17 0            NIBHEX 0
1204            *
1205            =tCOMMA EQU    #F1
1206 1EE18 000          CON(3)    0            F1  COMMA
1207 1EE1B 0000         REL(5) =TRMNTR
           0
1208 1EE20 0            NIBHEX 0
1209            *
1210            =tSEMIC EQU    #F2
1211 1EE21 000          CON(3)    0            F2  SEMICOLON
1212 1EE24 0000         REL(5) =TRMNTR
           0
1213 1EE29 0            NIBHEX 0
1214            *
1215            =tTO    EQU    #F3
1216 1EE2A 8E4          CON(3) 1256            F3  TO
1217 1EE2D 0000         NIBHEX 00000
           0
1218 1EE32 0            NIBHEX 0
1219            *
1220            =tTHEN  EQU    #F4
1221 1EE33 8B4          CON(3) 1208            F4  THEN
1222 1EE36 0000         REL(5) =TRMNTR
           0
1223 1EE3B 0            NIBHEX 0
1224            *
1225            =tELSE  EQU    #F5
1226 1EE3C 061          CON(3)  352            F5  ELSE
1227 1EE3F 0000         REL(5) =ELSE
           0
1228 1EE44 0            NIBHEX 0
1229            *
1230            =tSTEP  EQU    #F6
1231 1EE45 C74          CON(3) 1148            F6  STEP
1232 1EE48 0000         REL(5) =LABEL
           0
1233 1EE4D 0            NIBHEX 0
```

```
1234            *
1235            =tTAB   EQU    #F7
1236 1EE4E 6A4          CON(3) 1190            F7  TAB
1237 1EE51 0000         REL(5) =NXTSTM
         0
1238 1EE56 0            NIBHEX 0
1239            *
1240            =tALL   EQU    #F8
1241 1EE57 D10          CON(3)   29            F8  ALL
1242 1EE5A 0000         REL(5) =NXTSTM
         0
1243 1EE5F 0            NIBHEX 0
1244            *
1245            =tCALL  EQU    #F9
1246 1EE60 660          CON(3)  102            F9  CALL
1247 1EE63 0000         REL(5) =CALL
         0
1248 1EE68 D            NIBHEX D
1249            *
1250            =tCFLAG EQU    #FA
1251 1EE69 090          CON(3)  144            FA  CFLAG
1252 1EE6C 0000         REL(5) =CFLAG
         0
1253 1EE71 D            NIBHEX D
1254            *
1255            =tSFLAG EQU    #FB
1256 1EE72 C34          CON(3) 1084            FB  SFLAG
1257 1EE75 0000         REL(5) =SFLAG
         0
1258 1EE7A D            NIBHEX D
1259            *
1260            =t!     EQU    #FC
1261 1EE7B 000          CON(3)    0            FC  Comment
1262 1EE7E 0000         REL(5) =BANG
         0
1263 1EE83 0            NIBHEX 0
1264            *
1265            =tUSING EQU    #FD
1266 1EE84 D15          CON(3) 1309            FD  USING
1267 1EE87 0000         REL(5) =NXTSTM
         0
1268 1EE8C 0            NIBHEX 0
1269            *
1270            =tRUN   EQU    #FE
1271 1EE8D 824          CON(3) 1064            FE  RUN
1272 1EE90 0000         REL(5) =RUN
         0
1273 1EE95 D            NIBHEX D
1274            *
1275            =tIMAGE EQU    #FF
1276 1EE96 902          CON(3)  521            FF  IMAGE
1277 1EE99 0000         REL(5) =IMAGE
         0
1278 1EE9E 8            NIBHEX 8
1279            =tPRMEN EQU    #F8            F8  PRMEN   (End of Parm list-SUB)
```

```
1280          =tLBLST EQU   #F6          F6  Label Statement
1281          =t@     EQU   #F4          F4  @ (Continuation)
1282          =tEXTIF EQU   #F4          F4  Extended If
1283          =tPRMST EQU   #F3          F3  PRMST   (Start of Parm list-SU
1284          =tIN    EQU   #F2          F2  tIN     (for CALL)
1285          =tCOLON EQU   #E2          E2  HPIL colon token
1286          =tCVAL  EQU   #E1          E1  Call by value separator
1287          =tCREF  EQU   #E0          E0  Call by reference separator
1288          =tRFILE EQU   #DE          DE  Run file specified in RUNP
1289          =tLITRL EQU   #C4          C4  LITERAL (Literal label or file
1290          =LASTFN EQU   #B4          B4  Last Function
```

```
1291                    STITLE T e x t  T a b l e
1292            *  Text Table
1293 1EE9F      =LXTXTT
1294 1EE9F       TxTbSt                      Text table start
1295            *
1296 1EE9F 5            NIBHEX 5             ABS
1297 1EEA0 1424         NIBASC \ABS\
             35
1298 1EEA6 2A           NIBHEX 2A
1299            *
1300 1EEA8 7            NIBHEX 7             ACOS
1301 1EEA9 1434         NIBASC \ACOS\
             F435
1302 1EEB1 A9           NIBHEX A9
1303            *
1304 1EEB3 5            NIBHEX 5             ADD
1305 1EEB4 1444         NIBASC \ADD\
             44
1306 1EEBA 5D           NIBHEX 5D
1307            *
1308 1EEBC 5            NIBHEX 5             ALL
1309 1EEBD 14C4         NIBASC \ALL\
             C4
1310 1EEC3 8F           NIBHEX 8F
1311            *
1312 1EEC5 5            NIBHEX 5             AND
1313 1EEC6 14E4         NIBASC \AND\
             44
1314 1EECC B8           NIBHEX B8
1315            *
1316 1EECE 7            NIBHEX 7             ASIN
1317 1EECF 1435         NIBASC \ASIN\
             94E4
1318 1EED7 99           NIBHEX 99
1319            *
1320 1EED9 7            NIBHEX 7             ATAN
1321 1EEDA 1445         NIBASC \ATAN\
             14E4
1322 1EEE2 B9           NIBHEX B9
1323            *
1324 1EEE4 7            NIBHEX 7             AUTO
1325 1EEE5 1455         NIBASC \AUTO\
             45F4
1326 1EEED EE           NIBHEX EE
1327            *
1328 1EEEF 7            NIBHEX 7             BASE
1329 1EEF0 2414         NIBASC \BASE\
             3554
1330 1EEF8 9E           NIBHEX 9E
1331            *
1332 1EEFA 7            NIBHEX 7             BEEP
1333 1EEFB 2454         NIBASC \BEEP\
             5405
1334 1EF03 8E           NIBHEX 8E
1335            *
```

```
1336 1EF05 7          NIBHEX 7              CALL
1337 1EF06 3414       NIBASC \CALL\
           C4C4
1338 1EF0E 9F         NIBHEX 9F
1339           *
1340 1EF10 7          NIBHEX 7              CARD
1341 1EF11 3414       NIBASC \CARD\
           2544
1342 1EF19 0D         NIBHEX 0D
1343           *
1344 1EF1B 5          NIBHEX 5              CAT
1345 1EF1C 3414       NIBASC \CAT\
           45
1346 1EF22 CE         NIBHEX CE
1347           *
1348 1EF24 7          NIBHEX 7              CEIL
1349 1EF25 3454       NIBASC \CEIL\
           94C4
1350 1EF2D 27         NIBHEX 27
1351           *
1352 1EF2F 9          NIBHEX 9              CFLAG
1353 1EF30 3464       NIBASC \CFLAG\
           C414
           74
1354 1EF3A AF         NIBHEX AF
1355           *
1356 1EF3C 7          NIBHEX 7              CHR$
1357 1EF3D 3484       NIBASC \CHR$\
           2542
1358 1EF45 4A         NIBHEX 4A
1359           *
1360 1EF47 7          NIBHEX 7              COPY
1361 1EF48 34F4       NIBASC \COPY\
           0595
1362 1EF50 5B         NIBHEX 5B
1363           *
1364 1EF52 5          NIBHEX 5              COS
1365 1EF53 34F4       NIBASC \COS\
           35
1366 1EF59 79         NIBHEX 79
1367           *
1368 1EF5B 7          NIBHEX 7              DATA
1369 1EF5C 4414       NIBASC \DATA\
           4514
1370 1EF64 6C         NIBHEX 6C
1371           *
1372 1EF66 9          NIBHEX 9              DATE$
1373 1EF67 4414       NIBASC \DATE$\
           4554
           42
1374 1EF71 87         NIBHEX 87
1375           *
1376 1EF73 7          NIBHEX 7              DATE
1377 1EF74 4414       NIBASC \DATE\
           4554
```

```
1378 1EF7C 77          NIBHEX 77
1379          *
1380 1EF7E 5           NIBHEX 5              DEF
1381 1EF7F 4454        NIBASC \DEF\
          64
1382 1EF85 9B          NIBHEX 9B
1383          *
1384 1EF87 D           NIBHEX D              DEGREES
1385 1EF88 4454        NIBASC \DEGREES\
          7425
          5454
          35
1386 1EF96 3D          NIBHEX 3D
1387          *
1388 1EF98 5           NIBHEX 5              DEG
1389 1EF99 4454        NIBASC \DEG\
          74
1390 1EF9F F6          NIBHEX F6
1391          *
1392 1EFA1 9           NIBHEX 9              DELAY
1393 1EFA2 4454        NIBASC \DELAY\
          C414
          95
1394 1EFAC 6D          NIBHEX 6D
1395          *
1396 1EFAE B           NIBHEX B              DELETE
1397 1EFAF 4454        NIBASC \DELETE\
          C454
          4554
1398 1EFBB 7B          NIBHEX 7B
1399          *
1400 1EFBD D           NIBHEX D              DESTROY
1401 1EFBE 4454        NIBASC \DESTROY\
          3545
          25F4
          95
1402 1EFCC EB          NIBHEX EB
1403          *
1404 1EFCE 5           NIBHEX 5              DIM
1405 1EFCF 4494        NIBASC \DIM\
          D4
1406 1EFD5 CC          NIBHEX CC
1407          *
1408 1EFD7 7           NIBHEX 7              DISP
1409 1EFD8 4494        NIBASC \DISP\
          3505
1410 1EFE0 5C          NIBHEX 5C
1411          *
1412 1EFE2 5           NIBHEX 5              DIV
1413 1EFE3 4494        NIBASC \DIV\
          65
1414 1EFE9 68          NIBHEX 68
1415          *
1416 1EFEB 5           NIBHEX 5              DVZ
1417 1EFEC 4465        NIBASC \DVZ\
```

```
            A5
1418 1EFF2  1B          NIBHEX 1B
1419                *
1420 1EFF4  7           NIBHEX 7              EDIT
1421 1EFF5  5444        NIBASC \EDIT\
            9445
1422 1EFFD  8B          NIBHEX 8B
1423                *
1424 1EFFF  7           NIBHEX 7              ELSE
1425 1F000  54C4        NIBASC \ELSE\
            3554
1426 1F008  5F          NIBHEX 5F
1427                *
1428 1F00A  5           NIBHEX 5              END
1429 1F00B  54E4        NIBASC \END\
            44
1430 1F011  AD          NIBHEX AD
1431                *
1432 1F013  5           NIBHEX 5              EPS
1433 1F014  5405        NIBASC \EPS\
            35
1434 1F01A  17          NIBHEX 17
1435                *
1436 1F01C  7           NIBHEX 7              ERRL
1437 1F01D  5425        NIBASC \ERRL\
            25C4
1438 1F025  57          NIBHEX 57
1439                *
1440 1F027  7           NIBHEX 7              ERRN
1441 1F028  5425        NIBASC \ERRN\
            25E4
1442 1F030  67          NIBHEX 67
1443                *
1444 1F032  9           NIBHEX 9              ERROR
1445 1F033  5425        NIBASC \ERROR\
            25F4
            25
1446 1F03D  3E          NIBHEX 3E
1447                *
1448 1F03F  7           NIBHEX 7              EXOR
1449 1F040  5485        NIBASC \EXOR\
            F425
1450 1F048  C8          NIBHEX C8
1451                *
1452 1F04A  5           NIBHEX 5              EXP
1453 1F04B  5485        NIBASC \EXP\
            05
1454 1F051  49          NIBHEX 49
1455                *
1456 1F053  7           NIBHEX 7              FACT
1457 1F054  6414        NIBASC \FACT\
            3445
1458 1F05C  8A          NIBHEX 8A
1459                *
1460 1F05E  9           NIBHEX 9              FETCH
```

```
1461 1F05F 6454        NIBASC \FETCH\
           4534
           84
1462 1F069 8C          NIBHEX 8C
1463           *
1464 1F06B 3           NIBHEX 3              FN (lex only)
1465 1F06C 64E4        NIBASC \FN\
1466 1F070 00          NIBHEX 00
1467           *
1468 1F072 5           NIBHEX 5              FOR
1469 1F073 64F4        NIBASC \FOR\
           25
1470 1F079 3C          NIBHEX 3C
1471           *
1472 1F07B 3           NIBHEX 3              FP
1473 1F07C 6405        NIBASC \FP\
1474 1F080 B6          NIBHEX B6
1475           *
1476 1F082 9           NIBHEX 9              GOSUB
1477 1F083 74F4        NIBASC \GOSUB\
           3555
           24
1478 1F08D CD          NIBHEX CD
1479           *
1480 1F08F 7           NIBHEX 7              GOTO
1481 1F090 74F4        NIBASC \GOTO\
           45F4
1482 1F098 DD          NIBHEX DD
1483           *
1484 1F09A 3           NIBHEX 3              GO (lex only)
1485 1F09B 74F4        NIBASC \GO\
1486 1F09F 00          NIBHEX 00
1487           *
1488 1F0A1 3           NIBHEX 3              IF
1489 1F0A2 9464        NIBASC \IF\
1490 1F0A6 FD          NIBHEX FD
1491           *
1492 1F0A8 9           NIBHEX 9              IMAGE
1493 1F0A9 94D4        NIBASC \IMAGE\
           1474
           54
1494 1F0B3 FF          NIBHEX FF
1495           *
1496 1F0B5 5           NIBHEX 5              INF
1497 1F0B6 94E4        NIBASC \INF\
           64
1498 1F0BC 07          NIBHEX 07
1499           *
1500 1F0BE 9           NIBHEX 9              INPUT
1501 1F0BF 94E4        NIBASC \INPUT\
           0555
           45
1502 1F0C9 9C          NIBHEX 9C
1503           *
1504 1F0CB D           NIBHEX D              INTEGER
```

```
1505 1F0CC 94E4          NIBASC \INTEGER\
           4554
           7454
           25
1506 1F0DA AC            NIBHEX AC
1507             *
1508 1F0DC 5             NIBHEX 5                INT
1509 1F0DD 94E4          NIBASC \INT\
           45
1510 1F0E3 C9            NIBHEX C9
1511             *
1512 1F0E5 5             NIBHEX 5                INX
1513 1F0E6 94E4          NIBASC \INX\
           85
1514 1F0EC 2B            NIBHEX 2B
1515             *
1516 1F0EE 3             NIBHEX 3                IP
1517 1F0EF 9405          NIBASC \IP\
1518 1F0F3 A6            NIBHEX A6
1519             *
1520 1F0F5 3             NIBHEX 3                IS
1521 1F0F6 9435          NIBASC \IS\
1522 1F0FA 7E            NIBHEX 7E
1523             *
1524 1F0FC 5             NIBHEX 5                IVL
1525 1F0FD 9465          NIBASC \IVL\
           C4
1526 1F103 EA            NIBHEX EA
1527             *
1528 1F105 7             NIBHEX 7                KEY$
1529 1F106 B454          NIBASC \KEY$\
           9542
1530 1F10E 37            NIBHEX 37
1531             *
1532 1F110 7             NIBHEX 7                KEYS
1533 1F111 B454          NIBASC \KEYS\
           9535
1534 1F119 FC            NIBHEX FC
1535             *
1536 1F11B 5             NIBHEX 5                KEY
1537 1F11C B454          NIBASC \KEY\
           95
1538 1F122 5E            NIBHEX 5E
1539             *
1540 1F124 5             NIBHEX 5                LEN
1541 1F125 C454          NIBASC \LEN\
           E4
1542 1F12B 9A            NIBHEX 9A
1543             *
1544 1F12D 5             NIBHEX 5                LET
1545 1F12E C454          NIBASC \LET\
           45
1546 1F134 0C            NIBHEX 0C
1547             *
1548 1F136 B             NIBHEX B                LINPUT
```

```
1549 1F137 C494          NIBASC \LINPUT\
           E405
           5545
1550 1F143 FB            NIBHEX FB
1551            *
1552 1F145 7             NIBHEX 7                LIST
1553 1F146 C494          NIBASC \LIST\
           3545
1554 1F14E BB            NIBHEX BB
1555            *
1556 1F150 3             NIBHEX 3                LN
1557 1F151 C4E4          NIBASC \LN\
1558 1F155 19            NIBHEX 19
1559            *
1560 1F157 9             NIBHEX 9                LOG10
1561 1F158 C4F4          NIBASC \LOG10\
           7413
           03
1562 1F162 39            NIBHEX 39
1563            *
1564 1F164 5             NIBHEX 5                LOG
1565 1F165 C4F4          NIBASC \LOG\
           74
1566 1F16B 09            NIBHEX 09
1567            *
1568 1F16D 3             NIBHEX 3                LR
1569 1F16E C425          NIBASC \LR\
1570 1F172 6B            NIBHEX 6B
1571            *
1572 1F174 7             NIBHEX 7                MAIN
1573 1F175 D414          NIBASC \MAIN\
           94E4
1574 1F17D 2D            NIBHEX 2D
1575            *
1576 1F17F D             NIBHEX D                MAXREAL
1577 1F180 D414          NIBASC \MAXREAL\
           8525
           5414
           C4
1578 1F18E C6            NIBHEX C6
1579            *
1580 1F190 5             NIBHEX 5                MAX
1581 1F191 D414          NIBASC \MAX\
           85
1582 1F197 DA            NIBHEX DA
1583            *
1584 1F199 7             NIBHEX 7                MEAN
1585 1F19A D454          NIBASC \MEAN\
           14E4
1586 1F1A2 D9            NIBHEX D9
1587            *
1588 1F1A4 5             NIBHEX 5                MIN
1589 1F1A5 D494          NIBASC \MIN\
           E4
1590 1F1AB CA            NIBHEX CA
```

```
1591                *
1592 1F1AD 5              NIBHEX 5              MOD
1593 1F1AE D4F4           NIBASC \MOD\
           44
1594 1F1B4 47             NIBHEX 47
1595                *
1596 1F1B6 7              NIBHEX 7              NAME
1597 1F1B7 E414           NIBASC \NAME\
           D454
1598 1F1BF DB             NIBHEX DB
1599                *
1600 1F1C1 7              NIBHEX 7              NEXT
1601 1F1C2 E454           NIBASC \NEXT\
           8545
1602 1F1CA 4C             NIBHEX 4C
1603                *
1604 1F1CC 5              NIBHEX 5              NOT
1605 1F1CD E4F4           NIBASC \NOT\
           45
1606 1F1D3 18             NIBHEX 18
1607                *
1608 1F1D5 5              NIBHEX 5              NUM
1609 1F1D6 E455           NIBASC \NUM\
           D4
1610 1F1DC 3A             NIBHEX 3A
1611                *
1612 1F1DE 5              NIBHEX 5              OFF
1613 1F1DF F464           NIBASC \OFF\
           64
1614 1F1E5 1E             NIBHEX 1E
1615                *
1616 1F1E7 3              NIBHEX 3              ON
1617 1F1E8 F4E4           NIBASC \ON\
1618 1F1EC 0E             NIBHEX 0E
1619                *
1620 1F1EE B              NIBHEX B              OPTION
1621 1F1EF F405           NIBASC \OPTION\
           4594
           F4E4
1622 1F1FB DE             NIBHEX DE
1623                *
1624 1F1FD 3              NIBHEX 3              OR
1625 1F1FE F425           NIBASC \OR\
1626 1F202 D8             NIBHEX D8
1627                *
1628 1F204 5              NIBHEX 5              OVF
1629 1F205 F465           NIBASC \OVF\
           64
1630 1F20B FA             NIBHEX FA
1631                *
1632 1F20D 9              NIBHEX 9              PAUSE
1633 1F20E 0514           NIBASC \PAUSE\
           5535
           54
1634 1F218 7D             NIBHEX 7D
```

```
1635                *
1636 1F21A 3          NIBHEX 3              PI
1637 1F21B 0594       NIBASC \PI\
1638 1F21F 97         NIBHEX 97
1639                *
1640 1F221 7          NIBHEX 7              PORT
1641 1F222 05F4       NIBASC \PORT\
           2545
1642 1F22A 1D         NIBHEX 1D
1643                *
1644 1F22C 9          NIBHEX 9              PREDV
1645 1F22D 0525       NIBASC \PREDV\
           5444
           65
1646 1F237 F9         NIBHEX F9
1647                *
1648 1F239 9          NIBHEX 9              PRINT
1649 1F23A 0525       NIBASC \PRINT\
           94E4
           45
1650 1F244 DC         NIBHEX DC
1651                *
1652 1F246 9          NIBHEX 9              PURGE
1653 1F247 0555       NIBASC \PURGE\
           2574
           54
1654 1F251 BE         NIBHEX BE
1655                *
1656 1F253 D          NIBHEX D              RADIANS
1657 1F254 2514       NIBASC \RADIANS\
           4494
           14E4
           35
1658 1F262 4D         NIBHEX 4D
1659                *
1660 1F264 5          NIBHEX 5              RAD
1661 1F265 2514       NIBASC \RAD\
           44
1662 1F26B E6         NIBHEX E6
1663                *
1664 1F26D 7          NIBHEX 7              READ
1665 1F26E 2554       NIBASC \READ\
           1444
1666 1F276 7C         NIBHEX 7C
1667                *
1668 1F278 7          NIBHEX 7              REAL
1669 1F279 2554       NIBASC \REAL\
           14C4
1670 1F281 CB         NIBHEX CB
1671                *
1672 1F283 5          NIBHEX 5              REM
1673 1F284 2554       NIBASC \REM\
           D4
1674 1F28A 6E         NIBHEX 6E
1675                *
```

```
1676 1F28C D          NIBHEX D              RESTORE
1677 1F28D 2554       NIBASC \RESTORE\
           3545
           F425
           54
1678 1F29B ED         NIBHEX ED
1679           *
1680 1F29D 5          NIBHEX 5              RES
1681 1F29E 2554       NIBASC \RES\
           35
1682 1F2A4 F7         NIBHEX F7
1683           *
1684 1F2A6 B          NIBHEX B              RETURN
1685 1F2A7 2554       NIBASC \RETURN\
           4555
           25E4
1686 1F2B3 BD         NIBHEX BD
1687           *
1688 1F2B5 5          NIBHEX 5              RMD
1689 1F2B6 25D4       NIBASC \RMD\
           44
1690 1F2BC D6         NIBHEX D6
1691           *
1692 1F2BE 5          NIBHEX 5              RND
1693 1F2BF 25E4       NIBASC \RND\
           44
1694 1F2C5 0A         NIBHEX 0A
1695           *
1696 1F2C7 5          NIBHEX 5              RUN
1697 1F2C8 2555       NIBASC \RUN\
           E4
1698 1F2CE EF         NIBHEX EF
1699           *
1700 1F2D0 7          NIBHEX 7              SDEV
1701 1F2D1 3544       NIBASC \SDEV\
           5465
1702 1F2D9 E9         NIBHEX E9
1703           *
1704 1F2DB 9          NIBHEX 9              SFLAG
1705 1F2DC 3564       NIBASC \SFLAG\
           C414
           74
1706 1F2E6 BF         NIBHEX BF
1707           *
1708 1F2E8 5          NIBHEX 5              SGN
1709 1F2E9 3574       NIBASC \SGN\
           E4
1710 1F2EF 1A         NIBHEX 1A
1711           *
1712 1F2F1 9          NIBHEX 9              SHORT
1713 1F2F2 3584       NIBASC \SHORT\
           F425
           45
1714 1F2FC BC         NIBHEX BC
1715           *
```

```
1716 1F2FE 5          NIBHEX 5              SIN
1717 1F2FF 3594       NIBASC \SIN\
           E4
1718 1F305 69         NIBHEX 69
1719           *
1720 1F307 5          NIBHEX 5              SQR
1721 1F308 3515       NIBASC \SQR\
           25
1722 1F30E 29         NIBHEX 29
1723           *
1724 1F310 7          NIBHEX 7              STAT
1725 1F311 3545       NIBASC \STAT\
           1445
1726 1F319 EC         NIBHEX EC
1727           *
1728 1F31B 7          NIBHEX 7              STEP
1729 1F31C 3545       NIBASC \STEP\
           5405
1730 1F324 6F         NIBHEX 6F
1731           *
1732 1F326 7          NIBHEX 7              STOP
1733 1F327 3545       NIBASC \STOP\
           F405
1734 1F32F 9D         NIBHEX 9D
1735           *
1736 1F331 7          NIBHEX 7              STR$ (formerly VAL$)
1737 1F332 3545       NIBASC \STR$\
           2542
1738 1F33A 6A         NIBHEX 6A
1739           *
1740 1F33C 5          NIBHEX 5              SUB
1741 1F33D 3555       NIBASC \SUB\
           24
1742 1F343 1C         NIBHEX 1C
1743           *
1744 1F345 5          NIBHEX 5              TAB
1745 1F346 4514       NIBASC \TAB\
           24
1746 1F34C 7F         NIBHEX 7F
1747           *
1748 1F34E 5          NIBHEX 5              TAN
1749 1F34F 4514       NIBASC \TAN\
           E4
1750 1F355 89         NIBHEX 89
1751           *
1752 1F357 7          NIBHEX 7              THEN
1753 1F358 4584       NIBASC \THEN\
           54E4
1754 1F360 4F         NIBHEX 4F
1755           *
1756 1F362 9          NIBHEX 9              TIME$
1757 1F363 4594       NIBASC \TIME$\
           D454
           42
1758 1F36D 59         NIBHEX 59
```

```
1759                *
1760 1F36F 9           NIBHEX 9              TIMER
1761 1F370 4594        NIBASC \TIMER\
           D454
           25
1762 1F37A 4E          NIBHEX 4E
1763                *
1764 1F37C 7           NIBHEX 7              TIME
1765 1F37D 4594        NIBASC \TIME\
           D454
1766 1F385 B7          NIBHEX B7
1767                *
1768 1F387 3           NIBHEX 3              TO
1769 1F388 45F4        NIBASC \TO\
1770 1F38C 3F          NIBHEX 3F
1771                *
1772 1F38E 9           NIBHEX 9              TRACE
1773 1F38F 4525        NIBASC \TRACE\
           1434
           54
1774 1F399 AE          NIBHEX AE
1775                *
1776 1F39B 5           NIBHEX 5              UNF
1777 1F39C 55E4        NIBASC \UNF\
           64
1778 1F3A2 0B          NIBHEX 0B
1779                *
1780 1F3A4 9           NIBHEX 9              UPRC$
1781 1F3A5 5505        NIBASC \UPRC$\
           2534
           42
1782 1F3AF BA          NIBHEX BA
1783                *
1784 1F3B1 7           NIBHEX 7              USER
1785 1F3B2 5535        NIBASC \USER\
           5425
1786 1F3BA 2E          NIBHEX 2E
1787                *
1788 1F3BC 9           NIBHEX 9              USING
1789 1F3BD 5535        NIBASC \USING\
           94E4
           74
1790 1F3C7 DF          NIBHEX DF
1791                *
1792 1F3C9 5           NIBHEX 5              VAL
1793 1F3CA 6514        NIBASC \VAL\
           C4
1794 1F3D0 5A          NIBHEX 5A
1795                *
1796 1F3D2 7           NIBHEX 7              WAIT
1797 1F3D3 7514        NIBASC \WAIT\
           9445
1798 1F3DB 8D          NIBHEX 8D
1799 1F3DD 1FF   TxTbEn NIBHEX 1FF            Text termination
1800 1F3E0             END
```

ABS	Ext	-	812							
ACOS	Ext	-	772							
ADD	Ext	-	1067							
AND	Ext	-	699							
ARRAY	Ext	-	630							
ASIN	Ext	-	767							
ATAN	Ext	-	777							
AUTO	Ext	-	1192							
BANG	Ext	-	1262							
BEEP	Ext	-	1162							
BLDNUM	Ext	-	65	70	75	80	85	90	95	100
			105	110	115	144	149	154	159	164
			169	174	179	184	189	194	199	
CALL	Ext	-	1247							
CAT	Ext	-	1182							
CEIL	Ext	-	575							
CFLAG	Ext	-	1252							
CHR$	Ext	-	822							
CMPLX	Ext	-	615							
COMPAR	Ext	-	694							
CONCAT	Ext	-	689							
COPY	Ext	-	907							
COS	Ext	-	757							
D'LTE	Ext	-	917							
DATA	Ext	-	992							
DATE	Ext	-	600							
DATE$	Ext	-	605							
DEF	Ext	-	927							
DEG	Ext	-	560							
DEGREE	Ext	-	1057							
DELAY	Ext	-	1072							
DIM	Ext	-	1022							
DISP	Ext	-	987							
DIV	Ext	-	675							
DIVIDE	Ext	-	665							
DMARRY	Ext	-	635							
DSTROY	Ext	-	952							
DVZ	Ext	-	887							
DYNAMC	Ext	-	485	490	495	500	505	510	515	520
			525	530						
EDIT	Ext	-	922							
ELSE	Ext	-	1227							
END	Ext	-	1092							
ENDDEF	Ext	-	932							
ENDSUB	Ext	-	972							
EPS	Ext	-	570							
ERRL	Ext	-	590							
ERRN	Ext	-	595							
EXOR	Ext	-	704							
EXP	Ext	-	742							
EXPR	Ext	-	713	717						
FACT	Ext	-	842							
FETCH	Ext	-	1002							
FN	Ext	-	625							
FN-GO	Ext	-	56							

```
 FOR      Ext                  -   977
 FP       Ext                  -   540
 GOSUB    Ext                  -  1102
 GOTO     Ext                  -  1107
 IF       Ext                  -  1117
 IMAGE    Ext                  -  1277
 INF      Ext                  -   565
 INPUT    Ext                  -  1007
 INT      Ext                  -   782
 INTEGR   Ext                  -  1012
 INVLUT   Ext                  -   645
 INX      Ext                  -   892
 IP       Ext                  -   535
 IVL      Ext                  -   872
 KEY      Ext                  -  1147
 KEY$     Ext                  -   580
 LABEL    Ext                  -  1232
=LASTFN   Abs      180 #000B4  -  1290
 LEN      Ext                  -   847
 LET      Ext                  -   962
 LINPUT   Ext                  -   957
 LIST     Ext                  -   937
 LOG      Ext                  -   722   727
 LOG10    Ext                  -   737
 LPRP     Ext                  -   852
 LR       Ext                  -   912
=LXSPDT   Abs   124227 #1E543  -    20
=LXTXTT   Abs   126623 #1EE9F  -  1293
=MAINT    Abs   124319 #1E59F  -    53
=MAINTS   Abs   124215 #1E537  -    12
 MAX      Ext                  -   867
 MAXRL    Ext                  -   545
 MEAN     Ext                  -   787
 MIN      Ext                  -   862
 MINUS    Ext                  -   655
 MOD      Ext                  -   585
 MULTPY   Ext                  -   660
 NAME     Ext                  -   947
 NEXT     Ext                  -   982
 NOT      Ext                  -   650
 NUM      Ext                  -   817
 NXTSTM   Ext                  -  1137  1142  1157  1167  1237  1242  1267
 OFF      Ext                  -  1127
 ON       Ext                  -  1122
 ONEDGT   Ext                  -   282   287   292   297   302   307   312   317
                               322   327
 OPTION   Ext                  -  1187
 OR       Ext                  -   709
 OVF      Ext                  -   877
 PAUSE    Ext                  -  1077
 PERCNT   Ext                  -   670
 PI       Ext                  -   610
 PLUS     Ext                  -   680
 PREDV    Ext                  -   797
 PRINT    Ext                  -  1027
```

PURGE	Ext			-	1177							
RAD	Ext			-	555							
RADIAN	Ext			-	1062							
READ	Ext			-	997							
REAL	Ext			-	942							
REM	Ext			-	1152							
RES	Ext			-	640							
RESTOR	Ext			-	1112							
RETURN	Ext			-	1097							
RMD	Ext			-	550							
RND	Ext			-	802							
RTNSXM	Ext			-	50							
RUN	Ext			-	1272							
SDEV	Ext			-	792							
SFLAG	Ext			-	1257							
SGN	Ext			-	807							
SHORT	Ext			-	1017							
SIN	Ext			-	752							
SQR	Ext			-	732							
STAT	Ext			-	1032							
STATIC	Ext			-	359	363	367	371	375	379	383	387
					391	395	399	403	407	411	415	419
					423	427	431	435	439	443	447	451
					455	460						
STOP	Ext			-	1087							
STR$	Ext			-	832							
STRING	Ext			-	268							
STRLIT	Ext			-	221	243						
SUB	Ext			-	967							
SUB$	Ext			-	837							
TAN	Ext			-	762							
TIME	Ext			-	620							
TIME$	Ext			-	747							
TRACE	Ext			-	1172							
TRMNTR	Ext			-	60	119	124	129	134	139	203	207
					211	216	225	230	234	238	247	251
					255	259	263	273	277	331	335	339
					343	347	351	355	464	468	472	476
					480	1202	1207	1212	1222			
TxTbEn	Abs	127965	#1F3DD	-	1799	28	30	37	44	45	46	
TxTbSt	Abs	126623	#1EE9F	-	1294	28	30	37	44	45	46	48
UNF	Ext			-	882							
UPRC$	Ext			-	857							
USER	Ext			-	1132							
VAL	Ext			-	827							
WAIT	Ext			-	1082							
XFN	Ext			-	897	902						
XWORD	Ext			-	1197							
=a!	Abs	33	#00021	-	214							
=a"	Abs	34	#00022	-	219							
=a$	Abs	36	#00024	-	228							
=a'	Abs	39	#00027	-	241							
=a.	Abs	46	#0002E	-	271							
=a0	Abs	48	#00030	-	280							
=a1	Abs	49	#00031	-	285							

```
=a2       Abs      50 #00032 -   290
=a3       Abs      51 #00033 -   295
=a4       Abs      52 #00034 -   300
=a5       Abs      53 #00035 -   305
=a6       Abs      54 #00036 -   310
=a7       Abs      55 #00037 -   315
=a8       Abs      56 #00038 -   320
=a9       Abs      57 #00039 -   325
=t!       Abs     252 #000FC -  1260
=t%       Abs     133 #00085 -   668
=t&       Abs     137 #00089 -   687
=t*       Abs     131 #00083 -   658
=t+       Abs     135 #00087 -   678
=t-       Abs     130 #00082 -   653
=t/       Abs     132 #00084 -   663
=t@       Abs     244 #000F4 -  1281
=tABS     Abs     162 #000A2 -   810
=tACOS    Abs     154 #0009A -   770
=tADD     Abs     213 #000D5 -  1065
=tADIG0   Abs      96 #00060 -   483
=tADIG1   Abs      97 #00061 -   488
=tADIG2   Abs      98 #00062 -   493
=tADIG3   Abs      99 #00063 -   498
=tADIG4   Abs     100 #00064 -   503
=tADIG5   Abs     101 #00065 -   508
=tADIG6   Abs     102 #00066 -   513
=tADIG7   Abs     103 #00067 -   518
=tADIG8   Abs     104 #00068 -   523
=tADIG9   Abs     105 #00069 -   528
=tALL     Abs     248 #000F8 -  1240
=tAND     Abs     139 #0008B -   697
=tARRAY   Abs     125 #0007D -   628
=tASIN    Abs     153 #00099 -   765
=tATAN    Abs     155 #0009B -   775
=tAUTO    Abs     238 #000EE -  1190
=tBASE    Abs     233 #000E9 -  1165
=tBEEP    Abs     232 #000E8 -  1160
=tBIG     Abs      16 #00010 -   132
=tCALL    Abs     249 #000F9 -  1245
=tCARD    Abs     208 #000D0 -  1040
=tCAT     Abs     236 #000EC -  1180
=tCEIL    Abs     114 #00072 -   573
=tCFLAG   Abs     250 #000FA -  1250
=tCHR$    Abs     164 #000A4 -   820
=tCMPLX   Abs     122 #0007A -   613
=tCOLON   Abs     226 #000E2 -  1285
=tCOMMA   Abs     241 #000F1 -  1205
=tCOPY    Abs     181 #000B5 -   905
=tCOS     Abs     151 #00097 -   755
=tCREF    Abs     224 #000E0 -  1287
=tCVAL    Abs     225 #000E1 -  1286
=tDATA    Abs     198 #000C6 -   990
=tDATE    Abs     119 #00077 -   598
=tDATE$   Abs     120 #00078 -   603
=tDEF     Abs     185 #000B9 -   925
```

```
=tDEG    Abs     111 #0006F -   558
=tDEGRE  Abs     211 #000D3 -  1055
=tDELAY  Abs     214 #000D6 -  1070
=tDELET  Abs     183 #000B7 -   915
=tDIM    Abs     204 #000CC -  1020
=tDISP   Abs     197 #000C5 -   985
=tDIV    Abs     134 #00086 -   673
=tDMYAR  Abs     126 #0007E -   633
=tDSTRY  Abs     190 #000BE -   950
=tDVZ    Abs     177 #000B1 -   885
=tEDIT   Abs     184 #000B8 -   920
=tELSE   Abs     245 #000F5 -  1225
=tEND    Abs     218 #000DA -  1090
=tENDDF  Abs     186 #000BA -   930
=tENDSB  Abs     194 #000C2 -   970
=tEOL    Abs     240 #000F0 -  1200
=tEPS    Abs     113 #00071 -   568
=tERRL   Abs     117 #00075 -   588
=tERRN   Abs     118 #00076 -   593
=tERROR  Abs     227 #000E3 -  1135
=tEXOR   Abs     140 #0008C -   702
=tEXP    Abs     148 #00094 -   740
=tEXTIF  Abs     244 #000F4 -  1282
=tFACT   Abs     168 #000A8 -   840
=tFETCH  Abs     200 #000C8 -  1000
=tFFN    Abs     180 #000B4 -   900
=tFLT1   Abs      29 #0001D -   197
=tFLT10  Abs      20 #00014 -   152
=tFLT11  Abs      19 #00013 -   147
=tFLT12  Abs      18 #00012 -   142
=tFLT2   Abs      28 #0001C -   192
=tFLT3   Abs      27 #0001B -   187
=tFLT4   Abs      26 #0001A -   182
=tFLT5   Abs      25 #00019 -   177
=tFLT6   Abs      24 #00018 -   172
=tFLT7   Abs      23 #00017 -   167
=tFLT8   Abs      22 #00016 -   162
=tFLT9   Abs      21 #00015 -   157
=tFN     Abs     124 #0007C -   623
=tFOR    Abs     195 #000C3 -   975
=tFP     Abs     107 #0006B -   538
=tGOSUB  Abs     220 #000DC -  1100
=tGOTO   Abs     221 #000DD -  1105
=tIF     Abs     223 #000DF -  1115
=tIMAGE  Abs     255 #000FF -  1275
=tIN     Abs     242 #000F2 -  1284
=tINF    Abs     112 #00070 -   563
=tINPUT  Abs     201 #000C9 -  1005
=tINT    Abs     156 #0009C -   780
=tINT10  Abs       4 #00004 -    73
=tINT11  Abs       3 #00003 -    68
=tINT12  Abs       2 #00002 -    63
=tINT2   Abs      12 #0000C -   113
=tINT3   Abs      11 #0000B -   108
=tINT4   Abs      10 #0000A -   103
```

```
=tINT5    Abs       9 #00009 -    98
=tINT6    Abs       8 #00008 -    93
=tINT7    Abs       7 #00007 -    88
=tINT8    Abs       6 #00006 -    83
=tINT9    Abs       5 #00005 -    78
=tINTEG   Abs     202 #000CA - 1010
=tINX     Abs     178 #000B2 -   890
=tIP      Abs     106 #0006A -   533
=tIS      Abs     231 #000E7 - 1155
=tISUB$   Abs     167 #000A7 -   835
=tIVL     Abs     174 #000AE -   870
=tKEY     Abs     229 #000E5 - 1145
=tKEY$    Abs     115 #00073 -   578
=tKEYS    Abs     207 #000CF - 1035
=tLBLRF   Abs      14 #0000E -   122
=tLBLST   Abs     246 #000F6 - 1280
=tLEN     Abs     169 #000A9 -   845
=tLET     Abs     192 #000C0 -   960
=tLINE#   Abs      15 #0000F -   127
=tLINPT   Abs     191 #000BF -   955
=tLIST    Abs     187 #000BB -   935
=tLITRL   Abs     196 #000C4 - 1289
=tLN      Abs     145 #00091 -   725
=tLOG     Abs     144 #00090 -   720
=tLOG10   Abs     147 #00093 -   735
=tLPRP    Abs     170 #000AA -   850
=tLR      Abs     182 #000B6 -   910
=tMAIN    Abs     210 #000D2 - 1050
=tMAX     Abs     173 #000AD -   865
=tMAXRL   Abs     108 #0006C -   543
=tMEAN    Abs     157 #0009D -   785
=tMIN     Abs     172 #000AC -   860
=tMOD     Abs     116 #00074 -   583
=tNAME    Abs     189 #000BD -   945
=tNEXT    Abs     196 #000C4 -   980
=tNOT     Abs     129 #00081 -   648
=tNUM     Abs     163 #000A3 -   815
=tOFF     Abs     225 #000E1 - 1125
=tON      Abs     224 #000E0 - 1120
=tOPT'N   Abs     237 #000ED - 1185
=tOR      Abs     141 #0008D -   707
=tOVF     Abs     175 #000AF -   875
=tPAUSE   Abs     215 #000D7 - 1075
=tPI      Abs     121 #00079 -   608
=tPORT    Abs     209 #000D1 - 1045
=tPREDV   Abs     159 #0009F -   795
=tPRINT   Abs     205 #000CD - 1025
=tPRMEN   Abs     248 #000F8 - 1279
=tPRMST   Abs     243 #000F3 - 1283
=tPURGE   Abs     235 #000EB - 1175
=tRAD     Abs     110 #0006E -   553
=tRDIAN   Abs     212 #000D4 - 1060
=tREAD    Abs     199 #000C7 -   995
=tREAL    Abs     188 #000BC -   940
=tRELOP   Abs     138 #0008A -   692
```

```
=tREM     Abs     230 #000E6 -  1150
=tRES     Abs     127 #0007F -   638
=tRESTR   Abs     222 #000DE -  1110
=tRETRN   Abs     219 #000DB -  1095
=tRFILE   Abs     222 #000DE -  1288
=tRMD     Abs     109 #0006D -   548
=tRND     Abs     160 #000A0 -   800
=tRUN     Abs     254 #000FE -  1270
=tSDEV    Abs     158 #0009E -   790
=tSEMIC   Abs     242 #000F2 -  1210
=tSFLAG   Abs     251 #000FB -  1255
=tSGN     Abs     161 #000A1 -   805
=tSHORT   Abs     203 #000CB -  1015
=tSIN     Abs     150 #00096 -   750
=tSMALL   Abs      17 #00011 -   137
=tSQR     Abs     146 #00092 -   730
=tSTAT    Abs     206 #000CE -  1030
=tSTEP    Abs     246 #000F6 -  1230
=tSTOP    Abs     217 #000D9 -  1085
=tSTR$    Abs     166 #000A6 -   830
=tSUB     Abs     193 #000C1 -   965
=tSVAR    Abs      45 #0002D -   266
=tTAB     Abs     247 #000F7 -  1235
=tTAN     Abs     152 #00098 -   760
=tTHEN    Abs     244 #000F4 -  1220
=tTIME    Abs     123 #0007B -   618
=tTIME$   Abs     149 #00095 -   745
=tTIMER   Abs     228 #000E4 -  1140
=tTO      Abs     243 #000F3 -  1215
=tTRACE   Abs     234 #000EA -  1170
=tUNF     Abs     176 #000B0 -   880
=tUPRC$   Abs     171 #000AB -   855
=tUSER    Abs     226 #000E2 -  1130
=tUSING   Abs     253 #000FD -  1265
=tVAL     Abs     165 #000A5 -   825
=tWAIT    Abs     216 #000D8 -  1080
=tXFN     Abs     179 #000B3 -   895
=tXWORD   Abs     239 #000EF -  1195
=tZ       Abs      90 #0005A -   458
=t^       Abs     128 #00080 -   643
```

Input Parameters

  Source file name is SB&TAB::MS

  Listing file name is SB/TAB:TI:ML::-1

  Object file name is SB%TAB:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1                *    TTTTT  III   &     EEEEE  RRRR   M   M
 2                *      T     I   & &    E      R   R  MM MM
 3                *      T     I   & &    E      R   R  M M M
 4                *      T     I    &     EEEE   RRRR   M M M
 5                *      T     I   & & &  E      R R    M   M
 6                *      T     I   &  &   E      R  R   M   M
 7                *      T    III   && &  EEEEE  R   R  M   M
 8
 9                        TITLE  Mainframe Message Table
10 1F3E0                 ABS    #1F3E0
11                *
12 1F3E0 5425    =ERRMST NIBASC \ERR L\        Error message prefix.
         2502
         C4
13 1F3EA         =LEX:MN                       Address of mainframe LEX table.
14                *
15                =enull  EQU      0            (Null)
16 1F3EA 50              CON(2)   5
17 1F3EC 00              CON(2)   0            Message number   0
18 1F3EE C               CON(1)  12
19                *-------------------- Math messages ---------------------------
20                *
21                =eUNFLW EQU      1            Underflow        UN
22 1F3EF 81              CON(2)  24
23 1F3F1 10              CON(2)   1            Message number   1
24 1F3F3 8               CON(1)   8
25 1F3F4 55E6            NIBASC \Underflo\
         4656
         2766
         C6F6
26 1F404 77              NIBASC \w\
27 1F406 C               CON(1)  12
28                *
29                =eOVFLW EQU      2            Overflow         OV
30 1F407 61              CON(2)  22
31 1F409 20              CON(2)   2            Message number   2
32 1F40B 7               CON(1)   7
33 1F40C F467            NIBASC \Overflow\
         5627
         66C6
         F677
34 1F41C C               CON(1)  12
35                *
36                =eEXPO  EQU      3            EXPONENT(0)
37 1F41D C1              CON(2)  28
38 1F41F 30              CON(2)   3            Message number   3
39 1F421 A               CON(1)  10
40 1F422 5485            NIBASC \EXPONENT\
         05F4
         E454
         E445
41 1F432 8203            NIBASC \(0)\
         92
42 1F438 C               CON(1)  12
43                *
```

```
44                =eTNINF EQU      4          TAN=Inf
45 1F439 11               CON(2)  17
46 1F43B 40               CON(2)   4          Message number   4
47 1F43D 3                CON(1)   3
48 1F43E 4514             NIBASC \TAN=\
         E4D3
49 1F446 E                CON(1)  14
50 1F447 3F               CON(2) =eINF
51 1F449 C                CON(1)  12
52                *
53                =e0^NEG EQU      5          0^neg
54 1F44A 01               CON(2)  16
55 1F44C 50               CON(2)   5          Message number   5
56 1F44E 4                CON(1)   4
57 1F44F 03E5             NIBASC \0^neg\
         E656
         76
58 1F459 C                CON(1)  12
59                *
60                =e0^0   EQU      6          0^0
61 1F45A C0               CON(2)  12
62 1F45C 60               CON(2)   6          Message number   6
63 1F45E 2                CON(1)   2
64 1F45F 03E5             NIBASC \0^0\
         03
65 1F465 C                CON(1)  12
66                *
67                =eZRO/0 EQU      7          0/0
68 1F466 C0               CON(2)  12
69 1F468 70               CON(2)   7          Message number   7
70 1F46A 2                CON(1)   2
71 1F46B 03F2             NIBASC \0/0\
         03
72 1F471 C                CON(1)  12
73                *
74                =eZRDIV EQU      8          /Zero
75 1F472 01               CON(2)  16
76 1F474 80               CON(2)   8          Message number   8
77 1F476 4                CON(1)   4
78 1F477 F2A5             NIBASC \/Zero\
         5627
         F6
79 1F481 C                CON(1)  12
80                *
81                =eNEG^X EQU      9          Neg^Non-int
82 1F482 C1               CON(2)  28
83 1F484 90               CON(2)   9          Message number   9
84 1F486 A                CON(1)  10
85 1F487 E456             NIBASC \Neg^Non-\
         76E5
         E4F6
         E6D2
86 1F497 96E6             NIBASC \int\
         47
87 1F49D C                CON(1)  12
```

```
 88                *
 89                =eSQR-  EQU     10              SQR(neg)
 90 1F49E 61              CON(2)  22
 91 1F4A0 A0              CON(2)  10              Message number  10
 92 1F4A2 7               CON(1)   7
 93 1F4A3 3515            NIBASC \SQR(neg)\
          2582
          E656
          7692
 94 1F4B3 C               CON(1)  12
 95                *
 96                =eIVARG EQU     11              Invalid Arg
 97 1F4B4 F0              CON(2)  15
 98 1F4B6 B0              CON(2)  11              Message number  11
 99 1F4B8 E               CON(1)  14
100 1F4B9 CE              CON(2) =eINVLD
101 1F4BB 2               CON(1)   2
102 1F4BC 1427            NIBASC \Arg\
          76
103 1F4C2 C               CON(1)  12
104               *
105               =eLN0   EQU     12              LOG(0)
106 1F4C3 21              CON(2)  18
107 1F4C5 C0              CON(2)  12              Message number  12
108 1F4C7 5               CON(1)   5
109 1F4C8 C4F4            NIBASC \LOG(0)\
          7482
          0392
110 1F4D4 C               CON(1)  12
111               *
112               =eLOG-  EQU     13              LOG(neg)
113 1F4D5 61              CON(2)  22
114 1F4D7 D0              CON(2)  13              Message number  13
115 1F4D9 7               CON(1)   7
116 1F4DA C4F4            NIBASC \LOG(neg)\
          7482
          E656
          7692
117 1F4EA C               CON(1)  12
118               *
119               =eIF/IF EQU     14              Inf/Inf
120 1F4EB E0              CON(2)  14
121 1F4ED E0              CON(2)  14              Message number  14
122 1F4EF E               CON(1)  14
123 1F4F0 3F              CON(2) =eINF
124 1F4F2 0               CON(1)   0
125 1F4F3 F2              NIBASC \/\
126 1F4F5 E               CON(1)  14
127 1F4F6 3F              CON(2) =eINF
128 1F4F8 C               CON(1)  12
129               *
130               =eIF-IF EQU     15              Inf-Inf
131 1F4F9 E0              CON(2)  14
132 1F4FB F0              CON(2)  15              Message number  15
133 1F4FD E               CON(1)  14
```

```
134 1F4FE 3F          CON(2) =eINF
135 1F500 0           CON(1)   0
136 1F501 D2          NIBASC \-\
137 1F503 E           CON(1)  14
138 1F504 3F          CON(2) =eINF
139 1F506 C           CON(1)  12
140             *
141             =eIF*ZR EQU     16          Inf*0
142 1F507 D0          CON(2)  13
143 1F509 01          CON(2)  16          Message number  16
144 1F50B E           CON(1)  14
145 1F50C 3F          CON(2) =eINF
146 1F50E 1           CON(1)   1
147 1F50F A203        NIBASC \*0\
148 1F513 C           CON(1)  12
149             *
150             =e1^INF EQU     17          1^Inf
151 1F514 D0          CON(2)  13
152 1F516 11          CON(2)  17          Message number  17
153 1F518 1           CON(1)   1
154 1F519 13E5        NIBASC \1^\
155 1F51D E           CON(1)  14
156 1F51E 3F          CON(2) =eINF
157 1F520 C           CON(1)  12
158             *
159             =eINF^0 EQU     18          Inf^0
160 1F521 D0          CON(2)  13
161 1F523 21          CON(2)  18          Message number  18
162 1F525 E           CON(1)  14
163 1F526 3F          CON(2) =eINF
164 1F528 1           CON(1)   1
165 1F529 E503        NIBASC \^0\
166 1F52D C           CON(1)  12
167             *
168             =eSIGOP EQU     19          Signaled Op
169 1F52E C1          CON(2)  28
170 1F530 31          CON(2)  19          Message number  19
171 1F532 A           CON(1)  10
172 1F533 3596        NIBASC \Signaled\
          76E6
          16C6
          5646
173 1F543 02F4        NIBASC \ Op\
          07
174 1F549 C           CON(1)  12
175             *
176             =eUNORC EQU     20          Unordered
177 1F54A 81          CON(2)  24
178 1F54C 41          CON(2)  20          Message number  20
179 1F54E 8           CON(1)   8
180 1F54F 55E6        NIBASC \Unordere\
          F627
          4656
          2756
181 1F55F 46          NIBASC \d\
```

```
182 1F561 C           CON(1)  12
183               *
184               =eINX   EQU     21          Inexact          IX
185 1F562 41          CON(2)  20
186 1F564 51          CON(2)  21          Message number  21
187 1F566 6           CON(1)   6
188 1F567 94E6        NIBASC \Inexact\
          5687
          1636
          47
189 1F575 C           CON(1)  12
190               *---------------- System Errors ------------------------
191               *
192               =eLOBAT EQU     22          Low Battery
193 1F576 C1          CON(2)  28
194 1F578 61          CON(2)  22          Message number  22
195 1F57A A           CON(1)  10
196 1F57B C4F6        NIBASC \Low Batt\
          7702
          2416
          4747
197 1F58B 5627        NIBASC \ery\
          97
198 1F591 C           CON(1)  12
199               *
200               =eSYSER EQU     23          System Error
201 1F592 F1          CON(2)  31
202 1F594 71          CON(2)  23          Message number  23
203 1F596 B           CON(1)  11
204 1F597 B           CON(1)  11
205 1F598 3597        NIBASC \System E\
          3747
          56D6
          0254
206 1F5A8 2727        NIBASC \rror\
          F627
207 1F5B0 C           CON(1)  12
208               =eMMCOR EQU    eSYSER       Old symbolic for eSYSER.
209               *
210               *
211               =eMEM   EQU     24          Insufficient Memory
212 1F5B1 D2          CON(2)  45
213 1F5B3 81          CON(2)  24          Message number  24
214 1F5B5 A           CON(1)  10
215 1F5B6 94E6        NIBASC \Insuffic\
          3757
          6666
          9636
216 1F5C6 9656        NIBASC \ien\
          E6
217 1F5CC 7           CON(1)   7
218 1F5CD 4702        NIBASC \t Memory\
          D456
          D6F6
          2797
```

```
219 1F5DD C            CON(1)  12
220             *
221             =eMPI   EQU     25          Module Pulled
222 1F5DE C1           CON(2)  28
223 1F5E0 91           CON(2)  25          Message number  25
224 1F5E2 6            CON(1)   6
225 1F5E3 D4F6         NIBASC \Module \
          4657
          C656
          02
226 1F5F1 E            CON(1)  14
227 1F5F2 6F           CON(2) =ePULL
228 1F5F4 1            CON(1)   1
229 1F5F5 5646         NIBASC \ed\
230 1F5F9 C            CON(1)  12
231             *
232             =e2MROM EQU     26          Configuration
233 1F5FA 12           CON(2)  33
234 1F5FC A1           CON(2)  26          Message number  26
235 1F5FE B            CON(1)  11
236 1F5FF C            CON(1)  12
237 1F600 34F6         NIBASC \Configur\
          E666
          9676
          5727
238 1F610 1647         NIBASC \ation\
          96F6
          E6
239 1F61A C            CON(1)  12
240             *
241             =eAF    EQU     27          Invalid AF
242 1F61B DO           CON(2)  13
243 1F61D B1           CON(2)  27          Message number  27
244 1F61F E            CON(1)  14
245 1F620 CE           CON(2) =eINVLD
246 1F622 1            CON(1)   1
247 1F623 1464         NIBASC \AF\
248 1F627 C            CON(1)  12
249             *---------------- Program Errors ----------------------
250             *
251             =eSUBSC EQU     28          Subscript
252 1F628 81           CON(2)  24
253 1F62A C1           CON(2)  28          Message number  28
254 1F62C 8            CON(1)   8
255 1F62D 3557         NIBASC \Subscrip\
          2637
          3627
          9607
256 1F63D 47           NIBASC \t\
257 1F63F C            CON(1)  12
258             *
259             =eRECOR EQU     29          Record Ovfl
260 1F640 51           CON(2)  21
261 1F642 D1           CON(2)  29          Message number  29
262 1F644 5            CON(1)   5
```

```
263 1F645 2556          NIBASC \Record\
          36F6
          2746
264 1F651 E             CON(1)  14
265 1F652 5F            CON(2) =eOVFL*
266 1F654 C             CON(1)  12
267             *
268             =eSTMNF EQU     30          Stmt Not Found
269 1F655 11            CON(2)  17
270 1F657 E1            CON(2)  30          Message number  30
271 1F659 3             CON(1)   3
272 1F65A 3547          NIBASC \Stmt\
          D647
273 1F662 E             CON(1)  14
274 1F663 8E            CON(2) =eNFOUN
275 1F665 C             CON(1)  12
276             *
277             =eDATTY EQU     31          Data Type
278 1F666 81            CON(2)  24
279 1F668 F1            CON(2)  31          Message number  31
280 1F66A 8             CON(1)   8
281 1F66B 4416          NIBASC \Data Typ\
          4716
          0245
          9707
282 1F67B 56            NIBASC \e\
283 1F67D C             CON(1)  12
284             *
285             =eNODAT EQU     32          No Data
286 1F67E 41            CON(2)  20
287 1F680 02            CON(2)  32          Message number  32
288 1F682 6             CON(1)   6
289 1F683 E4F6          NIBASC \No Data\
          0244
          1647
          16
290 1F691 C             CON(1)  12
291             *
292             =eFNNtF EQU     33          FN Not Found
293 1F692 D0            CON(2)  13
294 1F694 12            CON(2)  33          Message number  33
295 1F696 1             CON(1)   1
296 1F697 64E4          NIBASC \FN\
297 1F69B E             CON(1)  14
298 1F69C 8E            CON(2) =eNFOUN
299 1F69E C             CON(1)  12
300             *
301             =eXFNNF EQU     34          XFN Not Found
302 1F69F B0            CON(2)  11
303 1F6A1 22            CON(2)  34          Message number  34
304 1F6A3 0             CON(1)   0
305 1F6A4 85            NIBASC \X\
306 1F6A6 E             CON(1)  14
307 1F6A7 12            CON(2) =eFNNtF
308 1F6A9 C             CON(1)  12
```

```
309                *
310                =eXWORD EQU     35          XWORD Not found
311 1F6AA 31              CON(2)  19
312 1F6AC 32              CON(2)  35          Message number  35
313 1F6AE 4               CON(1)   4
314 1F6AF 8575            NIBASC \XWORD\
          F425
          44
315 1F6B9 E               CON(1)  14
316 1F6BA 8E              CON(2) =eNFOUN
317 1F6BC C               CON(1)  12
318                *
319                =ePRMIS EQU     36          Parameter Mismatch
320 1F6BD B2              CON(2)  43
321 1F6BF 42              CON(2)  36          Message number  36
322 1F6C1 A               CON(1)  10
323 1F6C2 0516            NIBASC \Paramete\
          2716
          D656
          4756
324 1F6D2 2702            NIBASC \r M\
          D4
325 1F6D8 6               CON(1)   6
326 1F6D9 9637            NIBASC \ismatch\
          D616
          4736
          86
327 1F6E7 C               CON(1)  12
328                *
329                =eSTROV EQU     37          String Ovfl
330 1F6E8 51              CON(2)  21
331 1F6EA 52              CON(2)  37          Message number  37
332 1F6EC 5               CON(1)   5
333 1F6ED 3547            NIBASC \String\
          2796
          E676
334 1F6F9 E               CON(1)  14
335 1F6FA 5F              CON(2) =eOVFL*
336 1F6FC C               CON(1)  12
337                *
338                =eNUMIN EQU     38          Numeric Input
339 1F6FD 71              CON(2)  23
340 1F6FF 62              CON(2)  38          Message number  38
341 1F701 6               CON(1)   6
342 1F702 E457            NIBASC \Numeric\
          D656
          2796
          36
343 1F710 E               CON(1)  14
344 1F711 4F              CON(2) =eINPUT
345 1F713 C               CON(1)  12
346                *
347                =eTOOMI EQU     39          Too Many Inputs
348 1F714 71              CON(2)  23
349 1F716 72              CON(2)  39          Message number  39
```

```
350 1F718 E           CON(1)  14
351 1F719 FE          CON(2) =eTOO
352 1F71B 3           CON(1)   3
353 1F71C D416        NIBASC \Many\
          E697
354 1F724 E           CON(1)  14
355 1F725 4F          CON(2) =eINPUT
356 1F727 0           CON(1)   0
357 1F728 37          NIBASC \s\
358 1F72A C           CON(1)  12
359             *
360             =eTOOFI EQU     40          Too Few Inputs
361 1F72B 51          CON(2)  21
362 1F72D 82          CON(2)  40          Message number  40
363 1F72F E           CON(1)  14
364 1F730 FE          CON(2) =eTOO
365 1F732 2           CON(1)   2
366 1F733 6456        NIBASC \Few\
          77
367 1F739 E           CON(1)  14
368 1F73A 4F          CON(2) =eINPUT
369 1F73C 0           CON(1)   0
370 1F73D 37          NIBASC \s\
371 1F73F C           CON(1)  12
372             *
373             =eCHNL# EQU     41          Chnl# Not Found
374 1F740 31          CON(2)  19
375 1F742 92          CON(2)  41          Message number  41
376 1F744 4           CON(1)   4
377 1F745 3486        NIBASC \Chnl#\
          E6C6
          32
378 1F74F E           CON(1)  14
379 1F750 8E          CON(2) =eNFOUN
380 1F752 C           CON(1)  12
381             *
382             =eFwoNX EQU     42          FOR w/o NEXT
383 1F753 81          CON(2)  24
384 1F755 A2          CON(2)  42          Message number  42
385 1F757 2           CON(1)   2
386 1F758 64F4        NIBASC \FOR\
          25
387 1F75E E           CON(1)  14
388 1F75F BE          CON(2) =ew/o
389 1F761 3           CON(1)   3
390 1F762 E454        NIBASC \NEXT\
          8545
391 1F76A C           CON(1)  12
392             *
393             =eNXwoF EQU     43          NEXT w/o FOR
394 1F76B 81          CON(2)  24
395 1F76D B2          CON(2)  43          Message number  43
396 1F76F 3           CON(1)   3
397 1F770 E454        NIBASC \NEXT\
          8545
```

```
398 1F778 E             CON(1)  14
399 1F779 BE            CON(2) =eW/O
400 1F77B 2             CON(1)   2
401 1F77C 64F4          NIBASC \FOR\
          25
402 1F782 C             CON(1)  12
403             *
404             =eRWOGS EQU     44          RTN W/O GOSUB
405 1F783 A1            CON(2)  26
406 1F785 C2            CON(2)  44          Message number  44
407 1F787 2             CON(1)   2
408 1F788 2545          NIBASC \RTN\
          E4
409 1F78E E             CON(1)  14
410 1F78F BE            CON(2) =eW/O
411 1F791 4             CON(1)   4
412 1F792 74F4          NIBASC \GOSUB\
          3555
          24
413 1F79C C             CON(1)  12
414             *
415             =eINVIM EQU     45          Invalid IMAGE
416 1F79D 31            CON(2)  19
417 1F79F D2            CON(2)  45          Message number  45
418 1F7A1 E             CON(1)  14
419 1F7A2 CE            CON(2) =eINVLD
420 1F7A4 4             CON(1)   4
421 1F7A5 94D4          NIBASC \IMAGE\
          1474
          54
422 1F7AF C             CON(1)  12
423             *
424             =eINVUS EQU     46          Invalid USING
425 1F7B0 31            CON(2)  19
426 1F7B2 E2            CON(2)  46          Message number  46
427 1F7B4 E             CON(1)  14
428 1F7B5 CE            CON(2) =eINVLD
429 1F7B7 4             CON(1)   4
430 1F7B8 5535          NIBASC \USING\
          94E4
          74
431 1F7C2 C             CON(1)  12
432             *
433             =eIMGOV EQU     47          IMAGE Ovfl
434 1F7C3 31            CON(2)  19
435 1F7C5 F2            CON(2)  47          Message number  47
436 1F7C7 4             CON(1)   4
437 1F7C8 94D4          NIBASC \IMAGE\
          1474
          54
438 1F7D2 E             CON(1)  14
439 1F7D3 5F            CON(2) =eOVFL*
440 1F7D5 C             CON(1)  12
441             *
442             =eIVTAB EQU     48          Invalid TAB
```

```
443 1F7D6 F0          CON(2)  15
444 1F7D8 03          CON(2)  48          Message number  48
445 1F7DA E           CON(1)  14
446 1F7DB CE          CON(2) =eINVLD
447 1F7DD 2           CON(1)   2
448 1F7DE 4514        NIBASC \TAB\
          24
449 1F7E4 C           CON(1)  12
450           *
451           =eSPGNF EQU     49          Sub Not Found
452 1F7E5 F0          CON(2)  15
453 1F7E7 13          CON(2)  49          Message number  49
454 1F7E9 2           CON(1)   2
455 1F7EA 3557        NIBASC \Sub\
          26
456 1F7F0 E           CON(1)  14
457 1F7F1 8E          CON(2) =eNFOUN
458 1F7F3 C           CON(1)  12
459           *
460           =eVCNTX EQU     50          Var Context
461 1F7F4 11          CON(2)  17
462 1F7F6 23          CON(2)  50          Message number  50
463 1F7F8 3           CON(1)   3
464 1F7F9 6516        NIBASC \Var \
          2702
465 1F801 E           CON(1)  14
466 1F802 9E          CON(2) =eCNTXT
467 1F804 C           CON(1)  12
468           *
469           =eVARTY EQU    eVCNTX       Old symbolic for eVARTY.
470           *
471           =eIVSAR EQU     51          Invalid Stat Array
472 1F805 51          CON(2)  21
473 1F807 33          CON(2)  51          Message number  51
474 1F809 E           CON(1)  14
475 1F80A DE          CON(2) =eINVST
476 1F80C 5           CON(1)   5
477 1F80D 0214        NIBASC \ Array\
          2727
          1697
478 1F819 C           CON(1)  12
479           *
480           =eNVSTA EQU    eIVSAR       Old symbolic for eIVSAR.
481           =eNSVAR EQU    eIVSAR       Old symbolic for eIVSAR.
482           *
483           =eIVSTA EQU     52          Invalid Statistic
484 1F81A 31          CON(2)  19
485 1F81C 43          CON(2)  52          Message number  52
486 1F81E E           CON(1)  14
487 1F81F DE          CON(2) =eINVST
488 1F821 4           CON(1)   4
489 1F822 9637        NIBASC \istic\
          4796
          36
490 1F82C C           CON(1)  12
```

```
491                     *
492                     =eIVSOP EQU     53              Invalid Stat Op
493 1F82D F0                    CON(2)  15
494 1F82F 53                    CON(2)  53              Message number  53
495 1F831 E                     CON(1)  14
496 1F832 DE                    CON(2) =eINVST
497 1F834 2                     CON(1)   2
498 1F835 02F4                  NIBASC \ Op\
          07
499 1F83B C                     CON(1)  12
500                     *
501                     =eEOFIL EQU     54              End of File
502 1F83C 71                    CON(2)  23
503 1F83E 63                    CON(2)  54              Message number  54
504 1F840 6                     CON(1)   6
505 1F841 54E6                  NIBASC \End of \
          4602
          F666
          02
506 1F84F E                     CON(1)  14
507 1F850 AE                    CON(2) =eFILE
508 1F852 C                     CON(1)  12
509                     *
510                     =eILTFM EQU     55              Invalid Transform
511 1F853 B0                    CON(2)  11
512 1F855 73                    CON(2)  55              Message number  55
513 1F857 E                     CON(1)  14
514 1F858 CE                    CON(2) =eINVLD
515 1F85A E                     CON(1)  14
516 1F85B 1F                    CON(2) =eTFM
517 1F85D C                     CON(1)  12
518                     *
519                     =eTFFLD EQU     56              Transform Failed
520 1F85E 71                    CON(2)  23
521 1F860 83                    CON(2)  56              Message number  56
522 1F862 E                     CON(1)  14
523 1F863 1F                    CON(2) =eTFM
524 1F865 6                     CON(1)   6
525 1F866 0264                  NIBASC \ Failed\
          1696
          C656
          46
526 1F874 C                     CON(1)  12
527                     *------------------  File and Device Errors  ------------------
528                     *
529                     =eFnFND EQU     57              File Not Found
530 1F875 B0                    CON(2)  11
531 1F877 93                    CON(2)  57              Message number  57
532 1F879 E                     CON(1)  14
533 1F87A AE                    CON(2) =eFILE
534 1F87C E                     CON(1)  14
535 1F87D 8E                    CON(2) =eNFOUN
536 1F87F C                     CON(1)  12
537                     *
538                     =eFSPEC EQU     58              Invalid Filespec
```

```
539 1F880 41          CON(2)  20
540 1F882 A3          CON(2)  58          Message number  58
541 1F884 E           CON(1)  14
542 1F885 CE          CON(2) =eINVLD
543 1F887 E           CON(1)  14
544 1F888 AE          CON(2) =eFILE
545 1F88A 3           CON(1)   3
546 1F88B 3707        NIBASC \spec\
          5636
547 1F893 C           CON(1)  12
548           *
549           =eFEXST EQU     59          File Exists
550 1F894 71          CON(2)  23
551 1F896 B3          CON(2)  59          Message number  59
552 1F898 E           CON(1)  14
553 1F899 AE          CON(2) =eFILE
554 1F89B 6           CON(1)   6
555 1F89C 0254        NIBASC \ Exists\
          8796
          3747
          37
556 1F8AA C           CON(1)  12
557           *
558           =eFACCS EQU     60          Illegal Access
559 1F8AB 51          CON(2)  21
560 1F8AD C3          CON(2)  60          Message number  60
561 1F8AF E           CON(1)  14
562 1F8B0 6E          CON(2) =eILLEG
563 1F8B2 5           CON(1)   5
564 1F8B3 1436        NIBASC \Access\
          3656
          3737
565 1F8BF C           CON(1)  12
566           *
567           =eFPROT EQU     61          File Protect
568 1F8C0 B0          CON(2)  11
569 1F8C2 D3          CON(2)  61          Message number  61
570 1F8C4 E           CON(1)  14
571 1F8C5 AE          CON(2) =eFILE
572 1F8C7 E           CON(1)  14
573 1F8C8 8F          CON(2) =ePRTCT
574 1F8CA C           CON(1)  12
575           *
576           =eFOPEN EQU     62          File Open
577 1F8CB 31          CON(2)  19
578 1F8CD E3          CON(2)  62          Message number  62
579 1F8CF E           CON(1)  14
580 1F8D0 AE          CON(2) =eFILE
581 1F8D2 4           CON(1)   4
582 1F8D3 02F4        NIBASC \ Open\
          0756
          E6
583 1F8DD C           CON(1)  12
584           *
585           =eFTYPE EQU     63          Invalid File Type
```

```
586 1F8DE 61          CON(2)  22
587 1F8E0 F3          CON(2)  63          Message number  63
588 1F8E2 E           CON(1)  14
589 1F8E3 CE          CON(2) =eINVLD
590 1F8E5 E           CON(1)  14
591 1F8E6 AE          CON(2) =eFILE
592 1F8E8 4           CON(1)   4
593 1F8E9 0245        NIBASC \ Type\
          9707
          56
594 1F8F3 C           CON(1)  12
595             *
596             =eDVCNF EQU     64          Device Not Found
597 1F8F4 51          CON(2)  21
598 1F8F6 04          CON(2)  64          Message number  64
599 1F8F8 5           CON(1)   5
600 1F8F9 4456        NIBASC \Device\
          6796
          3656
601 1F905 E           CON(1)  14
602 1F906 8E          CON(2) =eNFOUN
603 1F908 C           CON(1)  12
604             *
605             =eL2LNG EQU     65          Line Too Long
606 1F909 C1          CON(2)  28
607 1F90B 14          CON(2)  65          Message number  65
608 1F90D 4           CON(1)   4
609 1F90E C496        NIBASC \Line \
          E656
          02
610 1F918 E           CON(1)  14
611 1F919 FE          CON(2) =eTOO
612 1F91B 3           CON(1)   3
613 1F91C C4F6        NIBASC \Long\
          E676
614 1F924 C           CON(1)  12
615             *--------------  Card Reader Errors  --------------------------
616             *
617             =ePROTD EQU     66          Write Protected
618 1F925 81          CON(2)  24
619 1F927 24          CON(2)  66          Message number  66
620 1F929 4           CON(1)   4
621 1F92A 7527        NIBASC \Write\
          9647
          56
622 1F934 E           CON(1)  14
623 1F935 8F          CON(2) =ePRTCT
624 1F937 1           CON(1)   1
625 1F938 5646        NIBASC \ed\
626 1F93C C           CON(1)  12
627             *
628             =eNOTIN EQU     67          Not This File
629 1F93D B1          CON(2)  27
630 1F93F 34          CON(2)  67          Message number  67
631 1F941 8           CON(1)   8
```

```
632 1F942 E4F6          NIBASC \Not This\
          4702
          4586
          9637
633 1F952 02            NIBASC \ \
634 1F954 E             CON(1)  14
635 1F955 AE            CON(2) =eFILE
636 1F957 C             CON(1)  12
637               *
638               =eVFYER EQU     68            Verify Fail
639 1F958 C1            CON(2)  28
640 1F95A 44            CON(2)  68            Message number  68
641 1F95C A             CON(1)  10
642 1F95D 6556          NIBASC \Verify F\
          2796
          6697
          0264
643 1F96D 1696          NIBASC \ail\
          C6
644 1F973 C             CON(1)  12
645               *
646               =eUNKCD EQU     69            Unknown Card
647 1F974 F1            CON(2)  31
648 1F976 54            CON(2)  69            Message number  69
649 1F978 B             CON(1)  11
650 1F979 B             CON(1)  11
651 1F97A 55E6          NIBASC \Unknown \
          B6E6
          F677
          E602
652 1F98A 3416          NIBASC \Card\
          2746
653 1F992 C             CON(1)  12
654               *
655               =eRWERR EQU     70            R/W Error
656 1F993 81            CON(2)  24
657 1F995 64            CON(2)  70            Message number  70
658 1F997 8             CON(1)   8
659 1F998 25F2          NIBASC \R/W Erro\
          7502
          5427
          27F6
660 1F9A8 27            NIBASC \r\
661 1F9AA C             CON(1)  12
662               *
663               =eTUFAS EQU     71            Too Fast
664 1F9AB 11            CON(2)  17
665 1F9AD 74            CON(2)  71            Message number  71
666 1F9AF E             CON(1)  14
667 1F9B0 FE            CON(2) =eTOO
668 1F9B2 3             CON(1)   3
669 1F9B3 6416          NIBASC \Fast\
          3747
670 1F9BB C             CON(1)  12
671               *
```

```
672                =eTUSLO EQU     72          Too Slow
673 1F9BC 11               CON(2)  17
674 1F9BE 84               CON(2)  72          Message number  72
675 1F9C0 E                CON(1)  14
676 1F9C1 FE               CON(2) =eTOO
677 1F9C3 3                CON(1)   3
678 1F9C4 35C6             NIBASC \Slow\
          F677
679 1F9CC C                CON(1)  12
680                *
681                =eWRGNM EQU     73          Wrong Name
682 1F9CD A1               CON(2)  26
683 1F9CF 94               CON(2)  73          Message number  73
684 1F9D1 9                CON(1)   9
685 1F9D2 7527             NIBASC \Wrong Na\
          F6E6
          7602
          E416
686 1F9E2 D656             NIBASC \me\
687 1F9E6 C                CON(1)  12
688                *
689                =eF2BIG EQU     74          File Too Big
690 1F9E7 51               CON(2)  21
691 1F9E9 A4               CON(2)  74          Message number  74
692 1F9EB E                CON(1)  14
693 1F9EC AE               CON(2) =eFILE
694 1F9EE 0                CON(1)   0
695 1F9EF 02               NIBASC \ \
696 1F9F1 E                CON(1)  14
697 1F9F2 FE               CON(2) =eTOO
698 1F9F4 2                CON(1)   2
699 1F9F5 2496             NIBASC \Big\
          76
700 1F9FB C                CON(1)  12
701                *------------------  Syntax Errors  --------------------------
702                *
703                =eSYNTX EQU     75          Syntax
704 1F9FC 21               CON(2)  18
705 1F9FE B4               CON(2)  75          Message number  75
706 1FA00 5                CON(1)   5
707 1FA01 3597             NIBASC \Syntax\
          E647
          1687
708 1FA0D C                CON(1)  12
709                *
710                =ePRNEX EQU     76          ) Expected
711 1FA0E B0               CON(2)  11
712 1FA10 C4               CON(2)  76          Message number  76
713 1FA12 0                CON(1)   0
714 1FA13 92               NIBASC \)\
715 1FA15 E                CON(1)  14
716 1FA16 7E               CON(2) =eEXPCT
717 1FA18 C                CON(1)  12
718                *
719                =eQUOEX EQU     77          Quote Expected
```

```
720 1FA19 31          CON(2)  19
721 1FA1B D4          CON(2)  77          Message number  77
722 1FA1D 4           CON(1)   4
723 1FA1E 1557        NIBASC \Quote\
          F647
          56
724 1FA28 E           CON(1)  14
725 1FA29 7E          CON(2) =eEXPCT
726 1FA2B C           CON(1)  12
727             *
728             =eEXCHR EQU     78          Excess Chars
729 1FA2C F1          CON(2)  31
730 1FA2E E4          CON(2)  78          Message number  78
731 1FA30 B           CON(1)  11
732 1FA31 B           CON(1)  11
733 1FA32 5487        NIBASC \Excess C\
          3656
          3737
          0234
734 1FA42 8616        NIBASC \hars\
          2737
735 1FA4A C           CON(1)  12
736             *
737             =eILCNT EQU     79          Illegal Context
738 1FA4B B0          CON(2)  11
739 1FA4D F4          CON(2)  79          Message number  79
740 1FA4F E           CON(1)  14
741 1FA50 6E          CON(2) =eILLEG
742 1FA52 E           CON(1)  14
743 1FA53 9E          CON(2) =eCNTXT
744 1FA55 C           CON(1)  12
745             *
746             =eILEXP EQU     80          Invalid Expr
747 1FA56 11          CON(2)  17
748 1FA58 05          CON(2)  80          Message number  80
749 1FA5A E           CON(1)  14
750 1FA5B CE          CON(2) =eINVLD
751 1FA5D 3           CON(1)   3
752 1FA5E 5487        NIBASC \Expr\
          0727
753 1FA66 C           CON(1)  12
754             *
755             =eILPAR EQU     81          Invalid Parm
756 1FA67 11          CON(2)  17
757 1FA69 15          CON(2)  81          Message number  81
758 1FA6B E           CON(1)  14
759 1FA6C CE          CON(2) =eINVLD
760 1FA6E 3           CON(1)   3
761 1FA6F 0516        NIBASC \Parm\
          27D6
762 1FA77 C           CON(1)  12
763             *
764             =eMSPAR EQU     82          Missing Parm
765 1FA78 F1          CON(2)  31
766 1FA7A 25          CON(2)  82          Message number  82
```

```
    767 1FA7C B           CON(1)  11
    768 1FA7D B           CON(1)  11
    769 1FA7E D496        NIBASC \Missing \
            3737
            96E6
            7602
    770 1FA8E 0516        NIBASC \Parm\
            27D6
    771 1FA96 C           CON(1)  12
    772           *
    773           =eILVAR EQU     83          Invalid Var
    774 1FA97 F0          CON(2)  15
    775 1FA99 35          CON(2)  83          Message number  83
    776 1FA9B E           CON(1)  14
    777 1FA9C CE          CON(2) =eINVLD
    778 1FA9E 2           CON(1)   2
    779 1FA9F 6516        NIBASC \Var\
            27
    780 1FAA5 C           CON(1)  12
    781           *
    782           =ePRCER EQU     84          Precedence
    783 1FAA6 A1          CON(2)  26
    784 1FAA8 45          CON(2)  84          Message number  84
    785 1FAAA 9           CON(1)   9
    786 1FAAB 0527        NIBASC \Preceden\
            5636
            5646
            56E6
    787 1FABB 3656        NIBASC \ce\
    788 1FABF C           CON(1)  12
    789           *
    790           =eILKEY EQU     85          Invalid Key
    791 1FAC0 F0          CON(2)  15
    792 1FAC2 55          CON(2)  85          Message number  85
    793 1FAC4 E           CON(1)  14
    794 1FAC5 CE          CON(2) =eINVLD
    795 1FAC7 2           CON(1)   2
    796 1FAC8 B456        NIBASC \Key\
            97
    797 1FACE C           CON(1)  12
    798           *
    799           =eROWRN EQU     86          Operand Expected
    800 1FACF 71          CON(2)  23
    801 1FAD1 65          CON(2)  86          Message number  86
    802 1FAD3 6           CON(1)   6
    803 1FAD4 F407        NIBASC \Operand\
            5627
            16E6
            46
    804 1FAE2 E           CON(1)  14
    805 1FAE3 7E          CON(2) =eEXPCT
    806 1FAE5 C           CON(1)  12
    807           *
    808           =eR1WRN EQU     87          Operator Expected
    809 1FAE6 91          CON(2)  25
```

```
    810 1FAE8 75          CON(2)  87          Message number  87
    811 1FAEA 7           CON(1)   7
    812 1FAEB F407        NIBASC \Operator\
              5627
              1647
              F627
    813 1FAFB E           CON(1)  14
    814 1FAFC 7E          CON(2) =eEXPCT
    815 1FAFE C           CON(1)  12
    816 1FAFF       =WRNMST                   Address of wrn msg prefix -13.
    817             *
    818             =eTFWRN EQU     88        TFM WRN L###: <msg>
    819 1FAFF F1          CON(2)  31
    820 1FB01 85          CON(2)  88          Message number  88
    821 1FB03 8           CON(1)   8
    822 1FB04 4564        NIBASC \TFM WRN \
              D402
              7525
              E402
    823 1FB14 C4          NIBASC \L\
    824 1FB16 F2          CON(2)  47
    825 1FB18 0           CON(1)   0
    826 1FB19 A3          NIBASC \:\
    827 1FB1B F1          CON(2)  31
    828 1FB1D C           CON(1)  12
    829             *--------------- Card Reader Messages  ------------------------
    830             *
    831             =ePLLC# EQU     89        Pull ### of ###
    832 1FB1E B0          CON(2)  11
    833 1FB20 95          CON(2)  89          Message number  89
    834 1FB22 E           CON(1)  14
    835 1FB23 6F          CON(2) =ePULL
    836 1FB25 E           CON(1)  14
    837 1FB26 7F          CON(2) =e#of#
    838 1FB28 C           CON(1)  12
    839             *
    840             =ePLLC  EQU     90        Pull Card
    841 1FB29 31          CON(2)  19
    842 1FB2B A5          CON(2)  90          Message number  90
    843 1FB2D E           CON(1)  14
    844 1FB2E 6F          CON(2) =ePULL
    845 1FB30 4           CON(1)   4
    846 1FB31 0234        NIBASC \ Card\
              1627
              46
    847 1FB3B C           CON(1)  12
    848             *
    849             =eWALGN EQU     91        Wrt: Align then ENDLN
    850 1FB3C F0          CON(2)  15
    851 1FB3E B5          CON(2)  91          Message number  91
    852 1FB40 2           CON(1)   2
    853 1FB41 7527        NIBASC \Wrt\
              47
    854 1FB47 E           CON(1)  14
    855 1FB48 0F          CON(2) =eALGN
```

```
856 1FB4A C             CON(1)  12
857             *
858             =eVALGN EQU     92          Vfy: Align then ENDLN
859 1FB4B F0            CON(2)  15
860 1FB4D C5            CON(2)  92          Message number  92
861 1FB4F 2             CON(1)   2
862 1FB50 6566          NIBASC \Vfy\
          97
863 1FB56 E             CON(1)  14
864 1FB57 0F            CON(2) =eALGN
865 1FB59 C             CON(1)  12
866             *
867             =eRALGN EQU     93          Read: Align then ENDLN
868 1FB5A 11            CON(2)  17
869 1FB5C D5            CON(2)  93          Message number  93
870 1FB5E 3             CON(1)   3
871 1FB5F 2556          NIBASC \Read\
          1646
872 1FB67 E             CON(1)  14
873 1FB68 0F            CON(2) =eALGN
874 1FB6A C             CON(1)  12
875             *
876             =ePALGN EQU     94          Prot: Align then ENDLN
877 1FB6B 11            CON(2)  17
878 1FB6D E5            CON(2)  94          Message number  94
879 1FB6F 3             CON(1)   3
880 1FB70 0527          NIBASC \Prot\
          F647
881 1FB78 E             CON(1)  14
882 1FB79 0F            CON(2) =eALGN
883 1FB7B C             CON(1)  12
884             *
885             =eUALGN EQU     95          Unpr: Align then ENDLN
886 1FB7C 11            CON(2)  17
887 1FB7E F5            CON(2)  95          Message number  95
888 1FB80 3             CON(1)   3
889 1FB81 55E6          NIBASC \Unpr\
          0727
890 1FB89 E             CON(1)  14
891 1FB8A 0F            CON(2) =eALGN
892 1FB8C C             CON(1)  12
893             *
894             =eCALGN EQU     96          Cat: Align then ENDLN
895 1FB8D F0            CON(2)  15
896 1FB8F 06            CON(2)  96          Message number  96
897 1FB91 2             CON(1)   2
898 1FB92 3416          NIBASC \Cat\
          47
899 1FB98 E             CON(1)  14
900 1FB99 0F            CON(2) =eALGN
901 1FB9B C             CON(1)  12
902             *
903             =eTRKDN EQU     97          Trk ### Done
904 1FB9C 91            CON(2)  25
905 1FB9E 16            CON(2)  97          Message number  97
```

```
    906 1FBA0 3           CON(1)   3
    907 1FBA1 4527        NIBASC \Trk \
              B602
    908 1FBA9 F3          CON(2)  63
    909 1FBAB 3           CON(1)   3
    910 1FBAC 44F6        NIBASC \Done\
              E656
    911 1FBB4 C           CON(1)  12
    912               *
    913               ************************************************************
    914               ************************************************************
    915               *
    916               **** Building Block words for messages.
    917               *
    918               *
    919               =eTRKOF EQU    229            (trk ### of ###)
    920 1FBB5 61          CON(2)  22
    921 1FBB7 5E          CON(2) 229            Message number 229
    922 1FBB9 4           CON(1)   4
    923 1FBBA 0282        NIBASC \ (trk\
              4727
              B6
    924 1FBC4 E           CON(1)  14
    925 1FBC5 7F          CON(2) =e#of#
    926 1FBC7 0           CON(1)   0
    927 1FBC8 92          NIBASC \)\
    928 1FBCA C           CON(1)  12
    929               *
    930               =eILLEG EQU    230            Illegal
    931 1FBCB 61          CON(2)  22
    932 1FBCD 6E          CON(2) 230            Message number 230
    933 1FBCF 7           CON(1)   7
    934 1FBD0 94C6        NIBASC \Illegal \
              C656
              7616
              C602
    935 1FBE0 C           CON(1)  12
    936               *
    937               =eEXPCT EQU    231            Expected
    938 1FBE1 81          CON(2)  24
    939 1FBE3 7E          CON(2) 231            Message number 231
    940 1FBE5 8           CON(1)   8
    941 1FBE6 0254        NIBASC \ Expecte\
              8707
              5636
              4756
    942 1FBF6 46          NIBASC \d\
    943 1FBF8 C           CON(1)  12
    944               *
    945               =eNFOUN EQU    232            Not Found
    946 1FBF9 A1          CON(2)  26
    947 1FBFB 8E          CON(2) 232            Message number 232
    948 1FBFD 9           CON(1)   9
    949 1FBFE 02E4        NIBASC \ Not Fou\
              F647
```

```
           0264
           F657
    950 1FC0E E646         NIBASC \nd\
    951 1FC12 C            CON(1)  12
    952             *
    953             =eCNTXT EQU    233          Context
    954 1FC13 41           CON(2)  20
    955 1FC15 9E           CON(2) 233           Message number 233
    956 1FC17 6            CON(1)   6
    957 1FC18 34F6         NIBASC \Context\
           E647
           5687
           47
    958 1FC26 C            CON(1)  12
    959             *
    960             =eFILE  EQU    234          File
    961 1FC27 E0           CON(2)  14
    962 1FC29 AE           CON(2) 234           Message number 234
    963 1FC2B 3            CON(1)   3
    964 1FC2C 6496         NIBASC \File\
           C656
    965 1FC34 C            CON(1)  12
    966             *
    967             =ew/o   EQU    235          w/o
    968 1FC35 01           CON(2)  16
    969 1FC37 BE           CON(2) 235           Message number 235
    970 1FC39 4            CON(1)   4
    971 1FC3A 0277         NIBASC \ w/o \
           F2F6
           02
    972 1FC44 C            CON(1)  12
    973             *
    974             =eINVLD EQU    236          Invalid
    975 1FC45 61           CON(2)  22
    976 1FC47 CE           CON(2) 236           Message number 236
    977 1FC49 7            CON(1)   7
    978 1FC4A 94E6         NIBASC \Invalid \
           6716
           C696
           4602
    979 1FC5A C            CON(1)  12
    980             *
    981             =eINVST EQU    237          Invalid Stat
    982 1FC5B 11           CON(2)  17
    983 1FC5D DE           CON(2) 237           Message number 237
    984 1FC5F E            CON(1)  14
    985 1FC60 CE           CON(2) =eINVLD
    986 1FC62 3            CON(1)   3
    987 1FC63 3547         NIBASC \Stat\
           1647
    988 1FC6B C            CON(1)  12
    989             *
    990             =eTOO   EQU    239          Too
    991 1FC6C E0           CON(2)  14
    992 1FC6E FE           CON(2) 239           Message number 239
```

```
  993 1FC70 3           CON(1)   3
  994 1FC71 45F6        NIBASC \Too \
            F602
  995 1FC79 C           CON(1)  12
  996             *
  997             =eALGN  EQU    240          : Align then ENDLN
  998 1FC7A B2          CON(2)  43
  999 1FC7C 0F          CON(2) 240           Message number 240
 1000 1FC7E A           CON(1)  10
 1001 1FC7F A302        NIBASC \: Align \
            14C6
            9676
            E602
 1002 1FC8F 4786        NIBASC \the\
            56
 1003 1FC95 6           CON(1)   6
 1004 1FC96 E602        NIBASC \n ENDLN\
            54E4
            44C4
            E4
 1005 1FCA4 C           CON(1)  12
 1006            *
 1007            =eTFM   EQU    241          Transform
 1008 1FCA5 81          CON(2)  24
 1009 1FCA7 1F          CON(2) 241           Message number 241
 1010 1FCA9 8           CON(1)   8
 1011 1FCAA 4527        NIBASC \Transfor\
            16E6
            3766
            F627
 1012 1FCBA D6          NIBASC \n\
 1013 1FCBC C           CON(1)  12
 1014            *
 1015            =eINF   EQU    243          Inf
 1016 1FCBD C0          CON(2)  12
 1017 1FCBF 3F          CON(2) 243           Message number 243
 1018 1FCC1 2           CON(1)   2
 1019 1FCC2 94E6        NIBASC \Inf\
            66
 1020 1FCC8 C           CON(1)  12
 1021            *
 1022            =eINPUT EQU    244          Input
 1023 1FCC9 21          CON(2)  18
 1024 1FCCB 4F          CON(2) 244           Message number 244
 1025 1FCCD 5           CON(1)   5
 1026 1FCCE 0294        NIBASC \ Input\
            E607
            5747
 1027 1FCDA C           CON(1)  12
 1028            *
 1029            =eOVFL* EQU    245          Ovfl
 1030 1FCDB 01          CON(2)  16
 1031 1FCDD 5F          CON(2) 245           Message number 245
 1032 1FCDF 4           CON(1)   4
 1033 1FCE0 02F4        NIBASC \ Ovfl\
```

```
                6766
                C6
1034 1FCEA C            CON(1)  12
1035            *
1036            =ePULL  EQU    246          Pull
1037 1FCEB E0           CON(2)  14
1038 1FCED 6F           CON(2) 246          Message number 246
1039 1FCEF 3            CON(1)   3
1040 1FCF0 0557         NIBASC \Pull\
                C6C6
1041 1FCF8 C            CON(1)  12
1042            *
1043            =e#of#  EQU    247          ### of ###
1044 1FCF9 31           CON(2)  19
1045 1FCFB 7F           CON(2) 247          Message number 247
1046 1FCFD 0            CON(1)   0
1047 1FCFE 02           NIBASC \ \
1048 1FD00 F3           CON(2)  63
1049 1FD02 2            CON(1)   2
1050 1FD03 F666         NIBASC \of \
                02
1051 1FD09 F2           CON(2)  47
1052 1FD0B C            CON(1)  12
1053            *
1054            =ePRTCT EQU    248          Protect
1055 1FD0C 61           CON(2)  22
1056 1FD0E 8F           CON(2) 248          Message number 248
1057 1FD10 7            CON(1)   7
1058 1FD11 0205         NIBASC \ Protect\
                27F6
                4756
                3647
1059 1FD21 C            CON(1)  12
1060            *
1061 1FD22 FF           NIBHEX FF           Table terminator
1062 1FD24              END
```

=ERRMST	Abs	127968	#1F3E0 -	12							
=LEX:MN	Abs	127978	#1F3EA -	13							
=WRNMST	Abs	129791	#1FAFF -	816							
=e#of#	Abs	247	#000F7 -	1043	837	925					
=e0^0	Abs	6	#00006 -	60							
=e0^NEG	Abs	5	#00005 -	53							
=e1^INF	Abs	17	#00011 -	150							
=e2MROM	Abs	26	#0001A -	232							
=eAF	Abs	27	#0001B -	241							
=eALGN	Abs	240	#000F0 -	997	855	864	873	882	891	900	
=eCALGN	Abs	96	#00060 -	894							
=eCHNL#	Abs	41	#00029 -	373							
=eCNTXT	Abs	233	#000E9 -	953	466	743					
=eDATTY	Abs	31	#0001F -	277							
=eDVCNF	Abs	64	#00040 -	596							
=eEOFIL	Abs	54	#00036 -	501							
=eEXCHR	Abs	78	#0004E -	728							
=eEXP0	Abs	3	#00003 -	36							
=eEXPCT	Abs	231	#000E7 -	937	716	725	805	814			
=eF2BIG	Abs	74	#0004A -	689							
=eFACCS	Abs	60	#0003C -	558							
=eFEXST	Abs	59	#0003B -	549							
=eFILE	Abs	234	#000EA -	960	507	533	544	553	571	580	591
				635	693						
=eFNNtF	Abs	33	#00021 -	292	307						
=eFOPEN	Abs	62	#0003E -	576							
=eFPROT	Abs	61	#0003D -	567							
=eFSPEC	Abs	58	#0003A -	538							
=eFTYPE	Abs	63	#0003F -	585							
=eFnFND	Abs	57	#00039 -	529							
=eFwoNX	Abs	42	#0002A -	382							
=eIF*ZR	Abs	16	#00010 -	141							
=eIF-IF	Abs	15	#0000F -	130							
=eIF/IF	Abs	14	#0000E -	119							
=eILCNT	Abs	79	#0004F -	737							
=eILEXP	Abs	80	#00050 -	746							
=eILKEY	Abs	85	#00055 -	790							
=eILLEG	Abs	230	#000E6 -	930	562	741					
=eILPAR	Abs	81	#00051 -	755							
=eILTFM	Abs	55	#00037 -	510							
=eILVAR	Abs	83	#00053 -	773							
=eIMGOV	Abs	47	#0002F -	433							
=eINF	Abs	243	#000F3 -	1015	50	123	127	134	138	145	156
				163							
=eINF^0	Abs	18	#00012 -	159							
=eINPUT	Abs	244	#000F4 -	1022	344	355	368				
=eINVIM	Abs	45	#0002D -	415							
=eINVLD	Abs	236	#000EC -	974	100	245	419	428	446	514	542
				589	750	759	777	794	985		
=eINVST	Abs	237	#000ED -	981	475	487	496				
=eINVUS	Abs	46	#0002E -	424							
=eINX	Abs	21	#00015 -	184							
=eIVARG	Abs	11	#0000B -	96							
=eIVSAR	Abs	51	#00033 -	471	480	481					
=eIVSOP	Abs	53	#00035 -	492							

```
=eIVSTA  Abs      52 #00034 -   483
=eIVTAB  Abs      48 #00030 -   442
=eL2LNG  Abs      65 #00041 -   605
=eLNO    Abs      12 #0000C -   105
=eLOBAT  Abs      22 #00016 -   192
=eLOG-   Abs      13 #0000D -   112
=eMEM    Abs      24 #00018 -   211
=eMMCOR  Abs      23 #00017 -   208
=eMPI    Abs      25 #00019 -   221
=eMSPAR  Abs      82 #00052 -   764
=eNEG^X  Abs       9 #00009 -    81
=eNFOUN  Abs     232 #000E8 -   945   274   298   316   379   457   535   602
=eNODAT  Abs      32 #00020 -   285
=eNOTIN  Abs      67 #00043 -   628
=eNSVAR  Abs      51 #00033 -   481
=eNUMIN  Abs      38 #00026 -   338
=eNVSTA  Abs      51 #00033 -   480
=eNXwoF  Abs      43 #0002B -   393
=eOVFL*  Abs     245 #000F5 -  1029   265   335   439
=eOVFLW  Abs       2 #00002 -    29
=ePALGN  Abs      94 #0005E -   876
=ePLLC   Abs      90 #0005A -   840
=ePLLC#  Abs      89 #00059 -   831
=ePRCER  Abs      84 #00054 -   782
=ePRMIS  Abs      36 #00024 -   319
=ePRNEX  Abs      76 #0004C -   710
=ePROTD  Abs      66 #00042 -   617
=ePRTCT  Abs     248 #000F8 -  1054   573   623
=ePULL   Abs     246 #000F6 -  1036   227   835   844
=eQUOEX  Abs      77 #0004D -   719
=eROWRN  Abs      86 #00056 -   799
=eR1WRN  Abs      87 #00057 -   808
=eRALGN  Abs      93 #0005D -   867
=eRECOR  Abs      29 #0001D -   259
=eRWERR  Abs      70 #00046 -   655
=eRwoGS  Abs      44 #0002C -   404
=eSIGOP  Abs      19 #00013 -   168
=eSPGNF  Abs      49 #00031 -   451
=eSQR-   Abs      10 #0000A -    89
=eSTMNF  Abs      30 #0001E -   268
=eSTROV  Abs      37 #00025 -   329
=eSUBSC  Abs      28 #0001C -   251
=eSYNTX  Abs      75 #0004B -   703
=eSYSER  Abs      23 #00017 -   200   208
=eTFFLD  Abs      56 #00038 -   519
=eTFM    Abs     241 #000F1 -  1007   516   523
=eTFWRN  Abs      88 #00058 -   818
=eTNINF  Abs       4 #00004 -    44
=eTOO    Abs     239 #000EF -   990   351   364   611   667   676   697
=eTOOFI  Abs      40 #00028 -   360
=eTOOMI  Abs      39 #00027 -   347
=eTRKDN  Abs      97 #00061 -   903
=eTRKOF  Abs     229 #000E5 -   919
=eTUFAS  Abs      71 #00047 -   663
=eTUSLO  Abs      72 #00048 -   672
```

```
=eUALGN  Abs      95 #0005F -   885
=eUNFLW  Abs       1 #00001 -    21
=eUNKCD  Abs      69 #00045 -   646
=eUNORC  Abs      20 #00014 -   176
=eVALGN  Abs      92 #0005C -   858
=eVARTY  Abs      50 #00032 -   469
=eVCNTX  Abs      50 #00032 -   460   469
=eVFYER  Abs      68 #00044 -   638
=eWALGN  Abs      91 #0005B -   849
=eWRGNM  Abs      73 #00049 -   681
=eXFNNF  Abs      34 #00022 -   301
=eXWORD  Abs      35 #00023 -   310
=eZRDIV  Abs       8 #00008 -    74
=eZRO/O  Abs       7 #00007 -    67
=enull   Abs       0 #00000 -    15
=ew/o    Abs     235 #000EB -   967   388   399   410
```

Input Parameters

  Source file name is TI&ERM::MS

  Listing file name is TI/ERM:TI:ML::-1

  Object file name is TI%ERM:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
1            *     SSS   BBBB    &     K   K   CCC   M   M
2            *    S   S  B   B  & &    K  K   C   C  MM MM
3            *    S      B   B  & &    K K    C      M M M
4            *     SSS   BBBB    &     KK     C      M M M
5            *        S  B   B  & & &  K K    C      M   M
6            *    S   S  B   B  &  &   K  K   C   C  M   M
7            *     SSS   BBBB    && &  K   K   CCC   M   M
8            *
9                    TITLE  Keycode Map<831212.1206>
10 1FD24             ABS    #1FD24
11
12           * The following keycodes are processed by KEYRD
13
14           =kcLC   EQU    1
15           =k#LC   EQU    106           Lowercase toggle
16
17           =kcUSER EQU    3
18           =k#USER EQU    109           Usermode toggle
19
20           =kcCTRL EQU    10
21           =k#CTRL EQU    158           CTRL prefix
22
23           =kcVIEW EQU    11
24           =k#VIEW EQU    110           VIEW prefix
25
26           =kcUSEX EQU    12
27           =k#USEX EQU    165           1USER
28
29           =kcLERR EQU    26
30           =k#LERR EQU    161           Last error message
31
32           * The following keycodes are processed by CHEDIT
33
34           =kc-CHR EQU    0
35           =k#-CHR EQU    104           Delete char
36
37           =kcI/R  EQU    2
38           =k#I/R  EQU    105           Insert/Replace toggle
39
40           =kc-LIN EQU    4
41           =k#-LIN EQU    107           Delete through EOL
42
43           =kcFLFT EQU    5
44           =k#FLFT EQU    159           Cursor far left
45
46           =kcFRT  EQU    6
47           =k#FRT  EQU    160           Cursor far right
48
49           =kcBKSP EQU    7
50           =k#BKSP EQU    103           Backspace
51
52           =kcLFT  EQU    8
53           =k#LFT  EQU    47            Cursor left
54
55           =kcRT   EQU    9
```

```
 56            =k#RT   EQU    48             Cursor right
 57
 58            * The following keycodes will terminate CHEDIT
 59
 60            =kcEOL  EQU    13
 61            =k#EOL  EQU    38             Endline
 62
 63            =kcATTN EQU    14
 64            =k#ATTN EQU    43             ATTN
 65
 66            =kcRUN  EQU    15
 67            =k#RUN  EQU    46             RUN
 68
 69            =kcCONT EQU    16
 70            =k#CONT EQU    112            CONT
 71
 72            =kcSST  EQU    17
 73            =k#SST  EQU    102            SST
 74
 75            =kcUP   EQU    18
 76            =k#UP   EQU    50             Up
 77
 78            =kcDOWN EQU    19
 79            =k#DOWN EQU    51             Down
 80
 81            =kcTOP  EQU    20
 82            =k#TOP  EQU    162            Top
 83
 84            =kcBOT  EQU    21
 85            =k#BOT  EQU    163            Bottom
 86
 87            =kcGON  EQU    22
 88            =k#GON  EQU    155            g-ON
 89
 90            =kcCALC EQU    23
 91            =k#CALC EQU    111            CALC
 92
 93            =kcOFF  EQU    24
 94            =k#OFF  EQU    99             OFF
 95
 96            =kcLAST EQU    25
 97            =k#LAST EQU    164            Command stack
 98
 99            *  The following are necessary key number definitions
100
101            =k#1    EQU    39
102            =k#2    EQU    40
103            =k#3    EQU    41
```

```
                        EJECT
                ********************************************************************
                ********************************************************************
                **
                ** Name:(S) KEYCOD  -  Keycode Map
                **
                ** Category:   KEYUTL
                **
                ** Purpose:    System keycode map.  Maps keys to their
                **      definition
                **
                ** Entry:
                **      Do not enter
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **  --------    ----------   ----------------------------------
                **  11/09/83    B.S.         Added documentation
                **
                ********************************************************************
                ********************************************************************
                =KEYCOD EQU    (*)-2
1FD24 15                NIBASC \Q\              001 01 ASCII Q
1FD26 75                NIBASC \W\              002 02 ASCII W
1FD28 54                NIBASC \E\              003 03 ASCII E
1FD2A 25                NIBASC \R\              004 04 ASCII R
1FD2C 45                NIBASC \T\              005 05 ASCII T
1FD2E 95                NIBASC \Y\              006 06 ASCII Y
1FD30 55                NIBASC \U\              007 07 ASCII U
1FD32 94                NIBASC \I\              008 08 ASCII I
1FD34 F4                NIBASC \O\              009 09 ASCII O
1FD36 05                NIBASC \P\              010 0A ASCII P
1FD38 73                NIBASC \7\              011 0B ASCII 7
1FD3A 83                NIBASC \8\              012 0C ASCII 8
1FD3C 93                NIBASC \9\              013 0D ASCII 9
1FD3E F2                NIBASC \/\              014 0E ASCII /
1FD40 14                NIBASC \A\              015 0F ASCII A
1FD42 35                NIBASC \S\              016 10 ASCII S
1FD44 44                NIBASC \D\              017 11 ASCII D
1FD46 64                NIBASC \F\              018 12 ASCII F
1FD48 74                NIBASC \G\              019 13 ASCII G
1FD4A 84                NIBASC \H\              020 14 ASCII H
1FD4C A4                NIBASC \J\              021 15 ASCII J
1FD4E B4                NIBASC \K\              022 16 ASCII K
1FD50 C4                NIBASC \L\              023 17 ASCII L
1FD52 D3                NIBASC \=\              024 18 ASCII =
1FD54 43                NIBASC \4\              025 19 ASCII 4
1FD56 53                NIBASC \5\              026 1A ASCII 5
1FD58 63                NIBASC \6\              027 1B ASCII 6
1FD5A A2                NIBASC \*\              028 1C ASCII *
1FD5C A5                NIBASC \Z\              029 1D ASCII Z
1FD5E 85                NIBASC \X\              030 1E ASCII X
1FD60 34                NIBASC \C\              031 1F ASCII C
1FD62 65                NIBASC \V\              032 20 ASCII V
```

```
159 1FD64 24          NIBASC \B\          033 21 ASCII B
160 1FD66 E4          NIBASC \N\          034 22 ASCII N
161 1FD68 D4          NIBASC \M\          035 23 ASCII M
162 1FD6A 82          NIBASC \(\          036 24 ASCII (
163 1FD6C 92          NIBASC \)\          037 25 ASCII )
164 1FD6E D0          CON(2) kcEOL        038 26 End Line key
165 1FD70 13          NIBASC \1\          039 27 ASCII 1
166 1FD72 23          NIBASC \2\          040 28 ASCII 2
167 1FD74 33          NIBASC \3\          041 29 ASCII 3
168 1FD76 D2          NIBASC \-\          042 2A ASCII -
169 1FD78 E0          CON(2) kcATTN       043 2B ON key
170 1FD7A             BSS    2            044 2C F shift key(Unused)
171 1FD7C             BSS    2            045 2D G shift key(Unused)
172 1FD7E F0          CON(2) kcRUN        046 2E RUN key
173 1FD80 80          CON(2) kcLFT        047 2F Cursor Left key
174 1FD82 90          CON(2) kcRT         048 30 Cursor Right key
175 1FD84 02          NIBASC \ \          049 31 ASCII Space
176 1FD86 21          CON(2) kcUP         050 32 Cursor Up key
177 1FD88 31          CON(2) kcDOWN       051 33 Cursor Down key
178 1FD8A D0          CON(2) kcEOL        052 34 End Line key(bottom half)
179 1FD8C 03          NIBASC \0\          053 35 ASCII 0
180 1FD8E E2          NIBASC \.\          054 36 ASCII .
181 1FD90 C2          NIBASC \,\          055 37 ASCII ,
182 1FD92 B2          NIBASC \+\          056 38 ASCII +
183                 *                           f SHIFTED KEYS
184 1FD94 00          CON(2) =tIF         057 39 Typing aid "IF "
185 1FD96 00          CON(2) =tTHEN       058 3A Typing aid " THEN "
186 1FD98 00          CON(2) =tELSE       059 3B Typing aid " ELSE "
187 1FD9A 00          CON(2) =tFOR        060 3C Typing aid "FOR "
188 1FD9C 00          CON(2) =tTO         061 3D Typing aid " TO "
189 1FD9E 00          CON(2) =tNEXT       062 3E Typing aid "NEXT "
190 1FDA0 00          CON(2) =tDEF        063 3F Typing aid "DEF "
191 1FDA2 00          CON(2) =tKEY        064 40 Typing aid "KEY "
192 1FDA4 00          CON(2) =tADD        065 41 Typing aid "ADD "
193 1FDA6 00          CON(2) =tLR         066 42 Typing aid "LR "
194 1FDA8 00          CON(2) =tPREDV      067 43 Typing aid "PREDV"
195 1FDAA 00          CON(2) =tMEAN       068 44 Typing aid "MEAN"
196 1FDAC 00          CON(2) =tSDEV       069 45 Typing aid "SDEV"
197 1FDAE 00          CON(2) =tSQR        070 46 Typing aid "SQR("
198 1FDB0 00          CON(2) =tCALL       071 47 Typing aid "CALL "
199 1FDB2 00          CON(2) =tGOSUB      072 48 Typing aid "GOSUB "
200 1FDB4 00          CON(2) =tRETRN      073 49 Typing aid "RETURN "
201 1FDB6 00          CON(2) =tGOTO       074 4A Typing aid "GOTO "
202 1FDB8 00          CON(2) =tINPUT      075 4B Typing aid "INPUT "
203 1FDBA 00          CON(2) =tPRINT      076 4C Typing aid "PRINT "
204 1FDBC 00          CON(2) =tDISP       077 4D Typing aid "DISP "
205 1FDBE 00          CON(2) =tDIM        078 4E Typing aid "DIM "
206 1FDC0 00          CON(2) =tBEEP       079 4F Typing aid "BEEP "
207 1FDC2 00          CON(2) =tFACT       080 50 Typing aid "FACT("
208 1FDC4 00          CON(2) =tSIN        081 51 Typing aid "SIN("
209 1FDC6 00          CON(2) =tCOS        082 52 Typing aid "COS("
210 1FDC8 00          CON(2) =tTAN        083 53 Typing aid "TAN("
211 1FDCA 00          CON(2) =tEXP        084 54 Typing aid "EXP("
212 1FDCC 00          CON(2) =tEDIT       085 55 Typing aid "EDIT "
213 1FDCE 00          CON(2) =tCAT        086 56 Typing aid "CAT "
```

```
214 1FDD0 00          CON(2) =tNAME       087 57 Typing aid "NAME "
215 1FDD2 00          CON(2) =tPURGE      088 58 Typing aid "PURGE "
216 1FDD4 00          CON(2) =tFETCH      089 59 Typing aid "FETCH "
217 1FDD6 00          CON(2) =tLIST       090 5A Typing aid "LIST "
218 1FDD8 00          CON(2) =tDELET      091 5B Typing aid "DELETE "
219 1FDDA 00          CON(2) =tAUTO       092 5C Typing aid "AUTO "
220 1FDDC 00          CON(2) =tCOPY       093 5D Typing aid "COPY "
221 1FDDE 00          CON(2) =tRES        094 5E Typing aid "RES"
222 1FDE0 00          CON(2) =tASIN       095 5F Typing aid "ASIN("
223 1FDE2 00          CON(2) =tACOS       096 60 Typing aid "ACOS("
224 1FDE4 00          CON(2) =tATAN       097 61 Typing aid "ATAN("
225 1FDE6 00          CON(2) =tLOG        098 62 Typing aid "LOG("
226 1FDE8 81          CON(2) kcOFF        099 63 OFF key
227 1FDEA             BSS    2            100 64 F shift key(Unused)
228 1FDEC             BSS    2            101 65 G shift key(Unused)
229 1FDEE 11          CON(2) kcSST        102 66 SST key
230 1FDF0 70          CON(2) kcBKSP       103 67 Back Space key
231 1FDF2 00          CON(2) kc-CHR       104 68 -CHAR key
232 1FDF4 20          CON(2) kcI/R        105 69 I/R key
233 1FDF6 10          CON(2) kcLC         106 6A Lower Case toggle
234 1FDF8 40          CON(2) kc-LIN       107 6B -LINE key
235 1FDFA 00          CON(2) =tRES        108 6C Typing aid "RES"
236                                               (Bottom half of key)
237 1FDFC 30          CON(2) kcUSER       109 6D USER mode toggle key
238 1FDFE B0          CON(2) kcVIEW       110 6E VIEW key
239 1FE00 71          CON(2) kcCALC       111 6F CALC mode key
240 1FE02 01          CON(2) kcCONT       112 70 CONT key
241              *                              g SHIFTED FUNCTIONS
242 1FE04 17          NIBASC \q\          113 71 ASCII q
243 1FE06 77          NIBASC \w\          114 72 ASCII w
244 1FE08 56          NIBASC \e\          115 73 ASCII e
245 1FE0A 27          NIBASC \r\          116 74 ASCII r
246 1FE0C 47          NIBASC \t\          117 75 ASCII t
247 1FE0E 97          NIBASC \y\          118 76 ASCII y
248 1FE10 57          NIBASC \u\          119 77 ASCII u
249 1FE12 96          NIBASC \i\          120 78 ASCII i
250 1FE14 F6          NIBASC \o\          121 79 ASCII o
251 1FE16 07          NIBASC \p\          122 7A ASCII p
252 1FE18 72          NIBASC \'\          123 7B ASCII '
253 1FE1A B7          NIBASC \{\          124 7C ASCII {
254 1FE1C D7          NIBASC \}\          125 7D ASCII }
255 1FE1E E5          NIBASC \^\          126 7E ASCII ^
256 1FE20 16          NIBASC \a\          127 7F ASCII a
257 1FE22 37          NIBASC \s\          128 80 ASCII s
258 1FE24 46          NIBASC \d\          129 81 ASCII d
259 1FE26 66          NIBASC \f\          130 82 ASCII f
260 1FE28 76          NIBASC \g\          131 83 ASCII g
261 1FE2A 86          NIBASC \h\          132 84 ASCII h
262 1FE2C A6          NIBASC \j\          133 85 ASCII j
263 1FE2E B6          NIBASC \k\          134 86 ASCII k
264 1FE30 C6          NIBASC \l\          135 87 ASCII l
265 1FE32 B3          NIBASC \;\          136 88 ASCII ;
266 1FE34 42          NIBASC \$\          137 89 ASCII $
267 1FE36 52          NIBASC \%\          138 8A ASCII %
268 1FE38 62          NIBASC \&\          139 8B ASCII &
```

```
269 1FE3A A3          NIBASC \:\          140 8C ASCII :
270 1FE3C A7          NIBASC \z\          141 8D ASCII z
271 1FE3E 87          NIBASC \x\          142 8E ASCII x
272 1FE40 36          NIBASC \c\          143 8F ASCII c
273 1FE42 67          NIBASC \v\          144 90 ASCII v
274 1FE44 26          NIBASC \b\          145 91 ASCII b
275 1FE46 E6          NIBASC \n\          146 92 ASCII n
276 1FE48 D6          NIBASC \m\          147 93 ASCII m
277 1FE4A B5          NIBASC \[\          148 94 ASCII [
278 1FE4C D5          NIBASC \]\          149 95 ASCII ]
279 1FE4E 91          CON(2) kcLAST       150 96 Command stack
280 1FE50 12          NIBASC \!\          151 97 ASCII !
281 1FE52 22          NIBASC \"\          152 98 ASCII "
282 1FE54 32          NIBASC \#\          153 99 ASCII #
283 1FE56 04          NIBASC \@\          154 9A ASCII @
284 1FE58 61          CON(2) kcGON        155 9B ON key
285 1FE5A             BSS    2            156 9C F shift key(Unused)
286 1FE5C             BSS    2            157 9D G shift key(Unused)
287 1FE5E A0          CON(2) kcCTRL       158 9E Control shift
288 1FE60 50          CON(2) kcFLFT       159 9F Cursor far left
289 1FE62 60          CON(2) kcFRT        160 A0 Cursor far right
290 1FE64 A1          CON(2) kcLERR       161 A1 ERRM (Last error message)
291 1FE66 41          CON(2) kcTOP        162 A2 Cursor to top
292 1FE68 51          CON(2) kcBOT        163 A3 Cursor to bottom
293 1FE6A 91          CON(2) kcLAST       164 A4 G End line key
294                                                (Bottom half of key)
295 1FE6C C0          CON(2) kcUSEX       165 A5 1USER
296 1FE6E C3          NIBASC \<\          166 A6 ASCII <
297 1FE70 E3          NIBASC \>\          167 A7 ASCII >
298 1FE72 F3          NIBASC \?\          168 A8 ASCII ?
299 1FE74             END
```

```
=KEYCOD  Abs  130338 #1FD22 -   126
=k#-CHR  Abs     104 #00068 -    35
=k#-LIN  Abs     107 #0006B -    41
=k#1     Abs      39 #00027 -   101
=k#2     Abs      40 #00028 -   102
=k#3     Abs      41 #00029 -   103
=k#ATTN  Abs      43 #0002B -    64
=k#BKSP  Abs     103 #00067 -    50
=k#BOT   Abs     163 #000A3 -    85
=k#CALC  Abs     111 #0006F -    91
=k#CONT  Abs     112 #00070 -    70
=k#CTRL  Abs     158 #0009E -    21
=k#DOWN  Abs      51 #00033 -    79
=k#EOL   Abs      38 #00026 -    61
=k#FLFT  Abs     159 #0009F -    44
=k#FRT   Abs     160 #000A0 -    47
=k#GON   Abs     155 #0009B -    88
=k#I/R   Abs     105 #00069 -    38
=k#LAST  Abs     164 #000A4 -    97
=k#LC    Abs     106 #0006A -    15
=k#LERR  Abs     161 #000A1 -    30
=k#LFT   Abs      47 #0002F -    53
=k#OFF   Abs      99 #00063 -    94
=k#RT    Abs      48 #00030 -    56
=k#RUN   Abs      46 #0002E -    67
=k#SST   Abs     102 #00066 -    73
=k#TOP   Abs     162 #000A2 -    82
=k#UP    Abs      50 #00032 -    76
=k#USER  Abs     109 #0006D -    18
=k#USEX  Abs     165 #000A5 -    27
=k#VIEW  Abs     110 #0006E -    24
=kc-CHR  Abs       0 #00000 -    34   231
=kc-LIN  Abs       4 #00004 -    40   234
=kcATTN  Abs      14 #0000E -    63   169
=kcBKSP  Abs       7 #00007 -    49   230
=kcBOT   Abs      21 #00015 -    84   292
=kcCALC  Abs      23 #00017 -    90   239
=kcCONT  Abs      16 #00010 -    69   240
=kcCTRL  Abs      10 #0000A -    20   287
=kcDOWN  Abs      19 #00013 -    78   177
=kcEOL   Abs      13 #0000D -    60   164   178
=kcFLFT  Abs       5 #00005 -    43   288
=kcFRT   Abs       6 #00006 -    46   289
=kcGON   Abs      22 #00016 -    87   284
=kcI/R   Abs       2 #00002 -    37   232
=kcLAST  Abs      25 #00019 -    96   279   293
=kcLC    Abs       1 #00001 -    14   233
=kcLERR  Abs      26 #0001A -    29   290
=kcLFT   Abs       8 #00008 -    52   173
=kcOFF   Abs      24 #00018 -    93   226
=kcRT    Abs       9 #00009 -    55   174
=kcRUN   Abs      15 #0000F -    66   172
=kcSST   Abs      17 #00011 -    72   229
=kcTOP   Abs      20 #00014 -    81   291
=kcUP    Abs      18 #00012 -    75   176
```

```
=kcUSER  Abs       3 #00003 -    17   237
=kcUSEX  Abs      12 #0000C -    26   295
=kcVIEW  Abs      11 #0000B -    23   238
 tACOS   Ext                 -   223
 tADD    Ext                 -   192
 tASIN   Ext                 -   222
 tATAN   Ext                 -   224
 tAUTO   Ext                 -   219
 tBEEP   Ext                 -   206
 tCALL   Ext                 -   198
 tCAT    Ext                 -   213
 tCOPY   Ext                 -   220
 tCOS    Ext                 -   209
 tDEF    Ext                 -   190
 tDELET  Ext                 -   218
 tDIM    Ext                 -   205
 tDISP   Ext                 -   204
 tEDIT   Ext                 -   212
 tELSE   Ext                 -   186
 tEXP    Ext                 -   211
 tFACT   Ext                 -   207
 tFETCH  Ext                 -   216
 tFOR    Ext                 -   187
 tGOSUB  Ext                 -   199
 tGOTO   Ext                 -   201
 tIF     Ext                 -   184
 tINPUT  Ext                 -   202
 tKEY    Ext                 -   191
 tLIST   Ext                 -   217
 tLOG    Ext                 -   225
 tLR     Ext                 -   193
 tMEAN   Ext                 -   195
 tNAME   Ext                 -   214
 tNEXT   Ext                 -   189
 tPREDV  Ext                 -   194
 tPRINT  Ext                 -   203
 tPURGE  Ext                 -   215
 tRES    Ext                 -   221   235
 tRETRN  Ext                 -   200
 tSDEV   Ext                 -   196
 tSIN    Ext                 -   208
 tSQR    Ext                 -   197
 tTAN    Ext                 -   210
 tTHEN   Ext                 -   185
 tTO     Ext                 -   188
```

Input Parameters

  Source file name is SB&KCM::MS

  Listing file name is SB/KCM:TI:ML::-1

  Object file name is SB%KCM:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
                        TITLE  Lextype Table <831212.1512>
1FE74                   ABS    #1FE74
                *      A    BBBB    &     L      X   X  TTTTT
                *     A A   B   B  & &    L      X   X    T
                *    A   A  B   B  & &    L       X X     T
                *    A   A  BBBB    &     L        X      T
                *    AAAAA  B   B  & & &  L       X X     T
                *    A   A  B   B  &  &   L      X   X    T
                *    A   A  BBBB    && &  LLLLL  X   X    T
                *
                **      Miscellaneous - type 0
                **      Letter        - type 1       A-Z
                **      Relational    - type 2       < = > ? #
                **      Numeric       - type 3       0-9 and decimal point

1FE74           =LXTYPT

1FE74 12                NIBASC \!\              BANG!
1FE76 0                 NIBHEX 0

1FE77 22                NIBASC \"\              QUOTE
1FE79 0                 NIBHEX 0

1FE7A 32                NIBASC \#\              HATCH (NOTEQUAL)
1FE7C 2                 NIBHEX 2

1FE7D 42                NIBASC \$\              DOLLAR
1FE7F 0                 NIBHEX 0

1FE80 00                CON(2) =t%              PERCENT
1FE82 0                 NIBHEX 0

1FE83 00                CON(2) =t&              AMPERSAND (CONCATENATE)
1FE85 0                 NIBHEX 0

1FE86 72                NIBASC \'\              APOSTROPHE
1FE88 0                 NIBHEX 0

1FE89 82                NIBASC \(\              LEFT PARENTHESIS
1FE8B 0                 NIBHEX 0

1FE8C 92                NIBASC \)\              RIGHT PARENTHESIS
1FE8E 0                 NIBHEX 0

1FE8F 00                CON(2) =t*              ASTERISK (MULTIPLY)
1FE91 0                 NIBHEX 0

1FE92 00                CON(2) =t+              PLUS
1FE94 0                 NIBHEX 0

1FE95 00                CON(2) =tCOMMA          COMMA
1FE97 0                 NIBHEX 0

1FE98 00                CON(2) =t-              MINUS
1FE9A 0                 NIBHEX 0
```

```
 56
 57 1FE9B E2          NIBASC \.\          PERIOD (DECIMAL POINT)
 58 1FE9D 3           NIBHEX 3
 59
 60 1FE9E 00          CON(2) =t/          SLASH (DIVIDE)
 61 1FEA0 0           NIBHEX 0
 62
 63 1FEA1 03          NIBASC \0\          ZERO
 64 1FEA3 3           NIBHEX 3
 65
 66 1FEA4 13          NIBASC \1\          ONE
 67 1FEA6 3           NIBHEX 3
 68
 69 1FEA7 23          NIBASC \2\          TWO
 70 1FEA9 3           NIBHEX 3
 71
 72 1FEAA 33          NIBASC \3\          THREE
 73 1FEAC 3           NIBHEX 3
 74
 75 1FEAD 43          NIBASC \4\          FOUR
 76 1FEAF 3           NIBHEX 3
 77
 78 1FEB0 53          NIBASC \5\          FIVE
 79 1FEB2 3           NIBHEX 3
 80
 81 1FEB3 63          NIBASC \6\          SIX
 82 1FEB5 3           NIBHEX 3
 83
 84 1FEB6 73          NIBASC \7\          SEVEN
 85 1FEB8 3           NIBHEX 3
 86
 87 1FEB9 83          NIBASC \8\          EIGHT
 88 1FEBB 3           NIBHEX 3
 89
 90 1FEBC 93          NIBASC \9\          NINE
 91 1FEBE 3           NIBHEX 3
 92
 93 1FEBF A3          NIBASC \:\          COLON
 94 1FEC1 0           NIBHEX 0
 95
 96 1FEC2 00          CON(2) =tSEMIC      SEMICOLON
 97 1FEC4 0           NIBHEX 0
 98
 99 1FEC5 C3          NIBASC \<\          LESSTHAN
100 1FEC7 2           NIBHEX 2
101
102 1FEC8 D3          NIBASC \=\          EQUAL
103 1FECA 2           NIBHEX 2
104
105 1FECB E3          NIBASC \>\          GREATERTHAN
106 1FECD 2           NIBHEX 2
107
108 1FECE F3          NIBASC \?\          INTERROGATIVE
109 1FED0 2           NIBHEX 2
110
```

```
111 1FED1 04          NIBASC \@\          @
112 1FED3 0           NIBHEX 0
113
114 1FED4 14          NIBASC \A\          A
115 1FED6 1           NIBHEX 1
116
117 1FED7 24          NIBASC \B\          B
118 1FED9 1           NIBHEX 1
119
120 1FEDA 34          NIBASC \C\          C
121 1FEDC 1           NIBHEX 1
122
123 1FEDD 44          NIBASC \D\          D
124 1FEDF 1           NIBHEX 1
125
126 1FEE0 54          NIBASC \E\          E
127 1FEE2 1           NIBHEX 1
128
129 1FEE3 64          NIBASC \F\          F
130 1FEE5 1           NIBHEX 1
131
132 1FEE6 74          NIBASC \G\          G
133 1FEE8 1           NIBHEX 1
134
135 1FEE9 84          NIBASC \H\          H
136 1FEEB 1           NIBHEX 1
137
138 1FEEC 94          NIBASC \I\          I
139 1FEEE 1           NIBHEX 1
140
141 1FEEF A4          NIBASC \J\          J
142 1FEF1 1           NIBHEX 1
143
144 1FEF2 B4          NIBASC \K\          K
145 1FEF4 1           NIBHEX 1
146
147 1FEF5 C4          NIBASC \L\          L
148 1FEF7 1           NIBHEX 1
149
150 1FEF8 D4          NIBASC \M\          M
151 1FEFA 1           NIBHEX 1
152
153 1FEFB E4          NIBASC \N\          N
154 1FEFD 1           NIBHEX 1
155
156 1FEFE F4          NIBASC \O\          O
157 1FF00 1           NIBHEX 1
158
159 1FF01 05          NIBASC \P\          P
160 1FF03 1           NIBHEX 1
161
162 1FF04 15          NIBASC \Q\          Q
163 1FF06 1           NIBHEX 1
164
165 1FF07 25          NIBASC \R\          R
```

```
166 1FF09 1           NIBHEX 1
167
168 1FF0A 35          NIBASC \S\          S
169 1FF0C 1           NIBHEX 1
170
171 1FF0D 45          NIBASC \T\          T
172 1FF0F 1           NIBHEX 1
173
174 1FF10 55          NIBASC \U\          U
175 1FF12 1           NIBHEX 1
176
177 1FF13 65          NIBASC \V\          V
178 1FF15 1           NIBHEX 1
179
180 1FF16 75          NIBASC \W\          W
181 1FF18 1           NIBHEX 1
182
183 1FF19 85          NIBASC \X\          X
184 1FF1B 1           NIBHEX 1
185
186 1FF1C 95          NIBASC \Y\          Y
187 1FF1E 1           NIBHEX 1
188
189 1FF1F A5          NIBASC \Z\          Z
190 1FF21 1           NIBHEX 1
191
192 1FF22 B5          NIBASC \[\          [
193 1FF24 0           NIBHEX 0
194
195 1FF25 00          CON(2) =tDIV        \
196 1FF27 0           NIBHEX 0
197
198 1FF28 D5          NIBASC \]\          ]
199 1FF2A 0           NIBHEX 0
200
201 1FF2B 00          CON(2) =t^          ^ (INVOLUTION)
202 1FF2D 0           NIBHEX 0
203
204 1FF2E             END
```

```
=LXTYPT  Abs  130676 #1FE74 -    16
 t%      Ext                -    30
 t&      Ext                -    33
 t*      Ext                -    45
 t+      Ext                -    48
 t-      Ext                -    54
 t/      Ext                -    60
 tCOMMA  Ext                -    51
 tDIV    Ext                -   195
 tSEMIC  Ext                -    96
 t^      Ext                -   201
```

Input Parameters

  Source file name is AB&LXT::MS

  Listing file name is AB/LXT:TI:ML::-1

  Object file name is AB%LXT:TI:MS::-1

```
                                    111111
                          0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
1            *     SSS    CCC    &     TTTTT    A    BBBB
2            *    S   S  C   C  & &      T     A A   B   B
3            *    S      C      & &      T    A   A  B   B
4            *     SSS   C       &       T    A   A  BBBB
5            *        S  C      & & &    T    AAAAA  B   B
6            *    S   S  C   C  &  &     T    A   A  B   B
7            *     SSS    CCC    && &    T    A   A  BBBB
8            *
9                     TITLE  File Type Table
10 1FF2E              ABS    #1FF2E
11
12
13           ********************************************************************
14           ********************************************************************
15           **
16           ** Name:    FTYPE   -  Mainframe File Type Table
17           **
18           ** Category:   FILUTL
19           **
20           ** Purpose:
21           **      This module contains the HP-71 File Type Table which
22           **      defines the file types the HP-71 mainframe recognizes.
23           **      The table starts at address =FTYPE.  This module does
24           **      not contain any executable code.
25           **
26           ** Entry:
27           **      None.
28           **
29           ** Exit:
30           **      None.
31           **
32           ** Calls:      None.
33           **
34           ** Uses.......
35           **  Inclusive: None.
36           **
37           ** Stk lvls:   None.
38           **
39           ** Detail:
40           **
41           **                  FILE TYPE TABLE FORMAT
42           **
43           **                    Size
44           **    Field         (nibs)                Meaning
45           ** -----------      ------   ----------------------------------
46           ** Create code         1     0: Normal mainframe file structure
47           **                            (BASIC, BIN, LEX, KEY, etc)
48           **                            File length measured in nibs,
49           **                            arbitrary format, subheaders
50           **                            allowed
51           **                           1: DATA file structure; up to 65535
52           **                            fixed length records of up to
53           **                            65535 bytes each; subheaders not
54           **                            allowed; file is initialized to
55           **                            FF's
```

```
                **                                  2: SDATA file structure; records are
                **                                     fixed length, 8 bytes each; file
                **                                     initialized to zeros; subheaders
                **                                     not allowed
                **                                  4: TEXT file structure; records are
                **                                     variable number of bytes; file
                **                                     initialized to FF's; subheaders
                **                                     are not allowed
                **                                  8: Special handler routine required
                **                                     to create this file; system will
                **                                     issue pCRT=8 poll
                **
                ** Copy code          1           0: Normal mainframe file structure;
                **                                     File can be copied into or out of
                **                                     HP-71 without aid from LEX file;
                **                                     Implementation Field contains
                **                                     file length on external copy,
                **                                     but is not present after file
                **                                     header when file is in memory
                **                                  1: DATA file structure; file can be
                **                                     copied into or out of HP-71 with
                **                                     no aid from LEX file; on external
                **                                     copy, the Implementation Field
                **                                     contains number of records and
                **                                     record length, and it is present
                **                                     immediately after file header
                **                                     when file is in memory
                **                                  2: SDATA file structure; file can be
                **                                     copied into or out of HP-71 with
                **                                     no aid from LEX file; on external
                **                                     copy, the Implementation Field
                **                                     contains number of records, but
                **                                     is not present after file header
                **                                     when file is in memory
                **                                  4: TEXT file structure; file can be
                **                                     copied into or out of HP-71 with
                **                                     no aid from LEX file; on external
                **                                     copy, the Implementation Field is
                **                                     zero, and it is not present after
                **                                     file header when file is in
                **                                     memory
                **                                  8: Special copy routine is required
                **                                     to copy file to or from HP-71;
                **                                     system will issue pWCRD8 poll;
                **                                     Implementation Field is present
                **                                     after file header when file is
                **                                     in memory
                **
                ** Execution code     1           1: File is executable (can be run)
                **                                  0: File is not executable
                **
                ** File data offset   2           Offset in nibs from start of file
                **                                  chain length field (in file header)
                **                                  to start of file data, skipping the
                **                                  Implementation Field, if present
```

```
111             **                          after file header (see Copy code,
112             **                          above), and also skipping the
113             **                          subheader (if any); this value is
114             **                          used to calculate the subheader
115             **                          length, when present
116             **
117             ** File type name    10      5 character ASCII name of the file
118             **                          type as displayed by CAT; padded
119             **                          with trailing blanks
120             **
121             ** Number of types    1      Number of file type numbers used
122             **                          by this type of file to indicate
123             **                          SECURE or PRIVATE states, if
124             **                          allowed; up to four type numbers
125             **                          may be used:
126             **
127             **                             Type
128             **                           Number    Protection Indicated
129             **                          --------   --------------------
130             **                           First       No protection
131             **                           Second      SECURE
132             **                           Third       PRIVATE
133             **                           Fourth      SECURE and PRIVATE
134             **
135             ** LIF type numbers   4     4 nibbles for each LIF type number
136             **                          indicated by the previous field
137             **
138             **
139             ********************************************************************
140             ********************************************************************
141             *
142 1FF2E       =FTYPE
143             *** DATA FILE (Interchange DATA File)
144 1FF2E 110           NIBHEX 110
145 1FF31 00            CON(2) =oDAsod
146 1FF33 4414          NIBASC \DATA \
            4514
            02
147 1FF3D 2             CON(1) 2
148 1FF3E 0000          CON(4) =fDATA
149 1FF42 0000          CON(4) (=fDATA)+1    Secure DATA file
150             *** BASIC FILE
151 1FF46 001           NIBHEX 001
152 1FF49 00            CON(2) =oBSsod
153 1FF4B 2414          NIBASC \BASIC\
            3594
            34
154 1FF55 4             CON(1) 4
155 1FF56 0000          CON(4) =fBASIC
156 1FF5A 0000          CON(4) (=fBASIC)+1   Secure BASIC
157 1FF5E 0000          CON(4) (=fBASIC)+2   Private BASIC
158 1FF62 0000          CON(4) (=fBASIC)+3   Secure, private BASIC
159             *** KEY FILE
160 1FF66 000           NIBHEX 000
161 1FF69 00            CON(2) =oKYsod
```

```
    162 1FF6B B454          NIBASC \KEY  \
                9502
                02
    163 1FF75 2             CON(1) 2
    164 1FF76 0000          CON(4) =fKEY
    165 1FF7A 0000          CON(4) (=fKEY)+1      Secure KEYS
    166               *** TEXT FILE
    167 1FF7E 440           NIBHEX 440
    168 1FF81 00            CON(2) =oTXsod
    169 1FF83 4554          NIBASC \TEXT \
                8545
                02
    170 1FF8D 2             CON(1) 2
    171 1FF8E 1000          CON(4) 1
    172 1FF92 1D0E          CON(4) #EOD1          Secure TEXT
    173               *** LIF1 FILE (same as TEXT)
    174 1FF96 440           NIBHEX 440
    175 1FF99 00            CON(2) =oTXsod
    176 1FF9B C494          NIBASC \LIF1 \
                6413
                02
    177 1FFA5 1             CON(1) 1
    178 1FFA6 1000          CON(4) 1
    179               *** SDATA FILE (Series 40 Data File)
    180 1FFAA 220           NIBHEX 220
    181 1FFAD 00            CON(2) =o41sod
    182 1FFAF 3544          NIBASC \SDATA\
                1445
                14
    183 1FFB9 1             CON(1) 1
    184 1FFBA 0D0E          CON(4) #EOD0
    185               *** BIN FILE (Binary File)
    186 1FFBE 001           NIBHEX 001
    187 1FFC1 00            CON(2) =oBNsod
    188 1FFC3 2494          NIBASC \BIN  \
                E402
                02
    189 1FFCD 4             CON(1) 4
    190 1FFCE 0000          CON(4) =fBIN
    191 1FFD2 0000          CON(4) (=fBIN)+1      Secure BIN
    192 1FFD6 0000          CON(4) (=fBIN)+2      Private BIN
    193 1FFDA 0000          CON(4) (=fBIN)+3      Secure, private BIN
    194               *** LEX FILE (Langauge Extension File)
    195 1FFDE 001           NIBHEX 001
    196 1FFE1 00            CON(2) =oLXsod
    197 1FFE3 C454          NIBASC \LEX  \
                8502
                02
    198 1FFED 4             CON(1) 4
    199 1FFEE 0000          CON(4) =fLEX
    200 1FFF2 0000          CON(4) (=fLEX)+1      Secure LEX
    201 1FFF6 0000          CON(4) (=fLEX)+2      Private LEX
    202 1FFFA 0000          CON(4) (=fLEX)+3      Secure, private LEX
    203               ****
    204               *
```

```
205 1FFFE FF          NIBHEX FF           TERMINATES TABLE
206           *
207 20000             END
```

```
=FTYPE   Abs  130862 #1FF2E -   142
 fBASIC  Ext                -   155   156   157   158
 fBIN    Ext                -   190   191   192   193
 fDATA   Ext                -   148   149
 fKEY    Ext                -   164   165
 fLEX    Ext                -   199   200   201   202
 o41sod  Ext                -   181
 oBNsod  Ext                -   187
 oBSsod  Ext                -   152
 oDAsod  Ext                -   145
 oKYsod  Ext                -   161
 oLXsod  Ext                -   196
 oTXsod  Ext                -   168   175
```

Input Parameters

  Source file name is SC&TAB::MS

  Listing file name is SC/TAB:TI:ML::-1

  Object file name is SC%TAB:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
 1            *     SSS   BBBB    &     RRRR     A    M   M
 2            *    S   S  B   B  & &    R   R   A A   MM MM
 3            *    S      B   B  & &    R   R  A   A  M M M
 4            *     SSS   BBBB    &     RRRR   A   A  M M M
 5            *        S  B   B  & & &  R R    AAAAA  M   M
 6            *    S   S  B   B  &  &   R  R   A   A  M   M
 7            *     SSS   BBBB    && &  R   R  A   A  M   M
 8            *
 9                    TITLE  System RAM Declarations<831212.1206>
10 2E100             ABS    #2E100
11            ********************************************************************
12            **                                                                **
13            **      S y s t e m   R A M   A l l o c a t i o n                 **
14            **                                                                **
15            ********************************************************************
16            *
17            * * *  Display driver addresses
18 2E100      =ANNAD1 BSS    1              Annunciator column 1
19 2E101      =ANN1.5 BSS    1
20 2E102      =ANNAD2 BSS    2              Annunciator column 2
21 2E104      =DD3ST  BSS    #2E160-*       Start of display driver 3
22 2E160      =DD3END BSS    #2E1F8-*       End of display driver 3
23 2E1F8      =TIMER3 BSS    #2E1FF-*       Timer 3
24 2E1FF      =DD3CTL BSS    1              Display driver 3 control nibble
25 2E200      =DD2ST  BSS    #2E260-*       Start of display driver 2
26 2E260      =DD2END BSS    #2E2F8-*       End of display driver 2
27 2E2F8      =TIMER2 BSS    #2E2FF-*       Timer 2
28 2E2FF      =DD2CTL BSS    1              Display driver 2 control nibble
29 2E300      =DD1ST  BSS    #2E34C-*       Start of display driver 1
30 2E34C      =DD1END                       End of display driver 1
31 2E34C      =ANNAD3 BSS    2              Annunciator column 3
32 2E34E      =ANNAD4 BSS    2              Annunciator column 4
33 2E350      =ROWDVR BSS    #2E3F8-*       Row Drivers
34 2E3F8      =TIMER1 BSS    #2E3FE-*       Timer 1
35 2E3FE      =DCONTR BSS    #2E3FF-*       Display contrast nibble
36 2E3FF      =DD1CTL BSS    #2F400-*       Display driver 1 control nibble
37            *
38            * * *  Start of interrupt RAM
39            *
40 2F400      =INTR4  BSS    16             (R4 and D0)
41 2F410      =INTA   BSS    16             (A reg)
42 2F420      =INTB   BSS    16             (B reg)
43 2F430      =INTM   BSS    8              Mode,Pointer,Carry,RSTK
44            *
45            * * *  End of interrupt RAM
46            *
47            =CMOSTV EQU    #168F          Value for CMOS test word
48 2F438      =CMOSTW BSS    4              CMOS test word
49 2F43C      =VECTOR BSS    5              Interrupt vector
50 2F441      =ATNDIS BSS    1              Attention disable flag
51 2F442      =OFFFLG
52 2F442      =ATNFLG BSS    1              Attention key hit flag
53 2F443      =KEYPTR BSS    1              Key buffer pointer
54 2F444      =KEYBUF BSS    15*2           Key buffer
55 2F462      =KEYSAV
```

```
 56 2F462      =KCOLD  BSS   1            14th column keymap(LSB=Bottom row)
 57 2F463      =KCOLC  BSS   1            13th
 58 2F464      =KCOLB  BSS   1            12th
 59 2F465      =KCOLA  BSS   1            11th
 60 2F466      =KCOL9  BSS   1            10th
 61 2F467      =KCOL8  BSS   1            9th
 62 2F468      =KCOL7  BSS   1            8th
 63 2F469      =KCOL6  BSS   1            7th
 64 2F46A      =KCOL5  BSS   1            6th
 65 2F46B      =KCOL4  BSS   1            5th
 66 2F46C      =KCOL3  BSS   1            4th
 67 2F46D      =KCOL2  BSS   1            3rd
 68 2F46E      =KCOL1  BSS   1            2nd
 69 2F46F      =KCOL0  BSS   1            1st
 70
 71 2F470      =DISINT BSS   1       Interrupt ignore flag
 72                                        used in keyscan
 73            *
 74            * The following memory is used by pseudo-device display driver
 75            *
 76 2F471      =WINDST BSS   2            Window start
 77 2F473      =WINDLN BSS   2            Window len
 78 2F475      =DSPSTA BSS   6            User status save, Dsp status save
 79 2F47B      =ESCSTA BSS   1            Escape status
 80 2F47C      =FIRSTC BSS   2            Buffer position of 1st chr in disp
 81 2F47E      =CURSOR BSS   2            Buffer position of cursor
 82 2F480      =DSPBFS BSS   2*96         96 character buffer (2 nibs/char)
 83 2F540      =DSPBFE
 84 2F540      =DSPMSK BSS   96/4         96 bits (4 bits/nib)
 85            *
 86            *  System Pointer Allocations
 87            *
 88 2F558      =MAINST BSS   5            Main Program Memory Start
 89            *
 90 2F55D      =UPD1ST                    Start of Update Addresses #1
 91 2F55D      =CURRST BSS   5            Current File Start
 92 2F562      =PRGMST BSS   5            Current Program Start
 93 2F567      =PRGMEN BSS   5            Current Program End
 94 2F56C      =CURREN BSS   5            Current File End
 95 2F571      =IOBFST                    Start of I/O Buffers
 96 2F571      =MAINEN BSS   5            Main Program Memory End
 97 2F576      =IOBFEN                    End of I/O Buffers
 98 2F576      =CLCBFR BSS   5            Calc Mode Pointers
 99 2F57B      =RFNBFR BSS   5
100 2F580      =RAWBFR BSS   5
101 2F585      =CLCSTK BSS   5            Calc Stack token stream start
102            *                           SYSEN,OUTBS,AVMEMS collapsed
103            *                           here by MAINLP
104 2F58A      =SYSEN  BSS   5            End of RAM used by System
105            *                           OUTBS and AVMEMS collapsed
106            *                           here at End of Parse,Decompile,
107            *                           TRANSFORM
108 2F58F      =OUTBS  BSS   5            Output Buffer Start
109            *                           Start of Output for Parse/Decomp
110 2F594      =AVMEMS BSS   5            Available Memory Start
```

```
111 2F599       =UPD1EN                     End of Update Addresses #1
112             *
113 2F599       =TASTK
114 2F599       =MTHSTK                     Arithmetic Stack
115 2F599       =AVMEME BSS    5            End of Available Memory
116             *                             (AVMEME collapsed to SAVSTK
117             *                               after statement execute)
118 2F59E       =SAVSTK                     Save Area Stack Pointer
119 2F59E       =TFORN
120 2F59E       =FORSTK BSS    5            FOR Stack
121 2F5A3       =TGSBS
122 2F5A3       =GSBSTK BSS    5            GOSUB Stack
123 2F5A8       =ACTIVE BSS    5            Active Variable Space
124 2F5AD       =CALSTK BSS    5            CALL Stack
125 2F5B2       =RAMEND BSS    5            End of Memory
126             *
127             * Variable Chain pointers
128             *
129 2F5B7       =PRMPTR BSS    7            Parameter Chain Pointer
130 2F5BE       =CHNLST                     Variable Chain Head Pointer List
131 2F5BE               BSS    26*7         26 Chains (7 nibbles per chain)
132             *
133             *
134             *   PCADDR through TMADR3 adjusted by RFADJ+
135             *   PCADDR through DATPTR saved by CALL
136             *   CNTADR through TMADR3 zeroed by CLRSTK/CLPSTK
137             *
138 2F674       =UPD2ST                     Start of Update Addresses #2
139 2F674       =DSPCHX BSS    5            Pointer to external display code
140 2F679       =PCADDR BSS    5            Program Counter @ Statement length
141 2F67E       =CNTADR BSS    5            Continue Address
142 2F683       =ERRSUB BSS    5            ON ERROR-GOSUB Return Address
143 2F688       =ERRADR BSS    5            ON ERROR Statement Address
144 2F68D       =ONINTR BSS    5            ON INTRPT Statement Address
145 2F692       =DATPTR BSS    5            DATA Statement Pointer
146 2F697       =TMRAD1 BSS    5            ON TIMER#1 Statement Address
147 2F69C       =TMRAD2 BSS    5            ON TIMER#2 Statement Address
148 2F6A1       =TMRAD3 BSS    5            ON TIMER#3 Statement Address
149 2F6A6       =UPD2EN                     End of Update Addresses #2
150             *
151             * Timer Intervals
152             *
153 2F6A6       =TMRIN1 BSS    8            TIMER#1 Interval
154 2F6AE       =TMRIN2 BSS    8            TIMER#2 Interval
155 2F6B6       =TMRIN3 BSS    8            TIMER#3 Interval
156             *
157 2F6BE       =STSAVE BSS    3            Status saved during Expr Execute
158 2F6C1       =LDCSPC BSS    5            Addr of space after line number
159             *
160 2F6C6       =INBS   BSS    5            Input buffer start
161 2F6CB       =AUTINC BSS    4            Increment value for AUTO
162 2F6CF       =LEXPTR BSS    5            Temporary storage for RESPTR
163 2F6D4       =CMDPTR                     Command stack pointer
164 2F6D4       =INADDR BSS    5            Stmt Len ptr for parse & decompile
165             *
```

```
                *  System and user flags
                *
                *     (Within each nibble, the lowest bit is the lowest
                *        absolute value flag number)
                *
2F6D9           =SYSFLG                          System flags
2F6D9                   BSS     1                Flags -1  ... -4
2F6DA                   BSS     1                Flags -5  ... -8
2F6DB                   BSS     1                Flags -9  ... -12
2F6DC                   BSS     1                Flags -13 ... -16
2F6DD                   BSS     1                Flags -17 ... -20
2F6DE                   BSS     1                Flags -21 ... -24
2F6DF                   BSS     1                Flags -25 ... -28
2F6E0                   BSS     1                Flags -29 ... -32
2F6E1                   BSS     1                Flags -33 ... -36
2F6E2                   BSS     1                Flags -37 ... -40
2F6E3                   BSS     1                Flags -41 ... -44
2F6E4                   BSS     1                Flags -45 ... -48
2F6E5                   BSS     1                Flags -49 ... -52
2F6E6                   BSS     1                Flags -53 ... -56
2F6E7                   BSS     1                Flags -57 ... -60
2F6E8                   BSS     1                Flags -61 ... -64
2F6E9           =FLGREG                          User flags
2F6E9                   BSS     1                Flags   0 ...  3
2F6EA                   BSS     1                Flags   4 ...  7
2F6EB                   BSS     1                Flags   8 ... 11
2F6EC                   BSS     1                Flags  12 ... 15
2F6ED                   BSS     1                Flags  16 ... 19
2F6EE                   BSS     1                Flags  20 ... 23
2F6EF                   BSS     1                Flags  24 ... 27
2F6F0                   BSS     1                Flags  28 ... 31
2F6F1                   BSS     1                Flags  32 ... 35
2F6F2                   BSS     1                Flags  36 ... 39
2F6F3                   BSS     1                Flags  40 ... 43
2F6F4                   BSS     1                Flags  44 ... 47
2F6F5                   BSS     1                Flags  48 ... 51
2F6F6                   BSS     1                Flags  52 ... 55
2F6F7                   BSS     1                Flags  56 ... 59
2F6F8                   BSS     1                Flags  60 ... 63
                =DSPFMT EQU     (FLGREG)-13      STD=0,FIX=1,SCI=2,ENG=3
                =DSPDGT EQU     (FLGREG)-12
2F6F9           =TRPREG                          IEEE exception traps
2F6F9           =INXNIB BSS     1                Inexact result trap
2F6FA           =UNFNIB BSS     1                Underflow trap
2F6FB           =OVFNIB BSS     1                Overflow trap
2F6FC           =DVZNIB BSS     1                Divide by zero trap
2F6FD           =IVLNIB BSS     1                Invalid result trap
                *
                *  Random Number Seed
                *
2F6FE           =RNSEED BSS     15
                *
                *  Alarm Clock System RAM
                *
                *
```

```
    221 2F70D       =NXTIRQ BSS    12              Time of next SREQ
    222 2F719       =ALRM1  BSS    12              ON TIMER #1
    223 2F725       =ALRM2  BSS    12              ON TIMER #2
    224 2F731       =ALRM3  BSS    12              ON TIMER #3
    225 2F73D       =ALRM4  BSS    12              Timeout
    226 2F749       =ALRM5  BSS    12              Pause
    227 2F755       =ALRM6  BSS    12              External alarm
    228 2F761       =PNDALM BSS    2               Bitmap of pending alarms
    229             *
    230             *   Storage needed for accuracy factor stuff
    231             *
    232 2F763       =TIMOFS BSS    12              Time error offset
    233 2F76F       =TIMLST BSS    12              Time last set
    234 2F77B       =TIMLAF BSS    12              Time of last AF correction
    235 2F787       =TIMAF  BSS    6               Accuracy Factor
    236             *
    237 2F78D       =IS-TBL                        Table of "IS" assignments
    238 2F78D       =IS-DSP BSS    7
    239 2F794       =IS-PRT BSS    7
    240 2F79B       =IS-INP BSS    7
    241 2F7A2       =IS-PLT BSS    7
    242             *
    243 2F7A9       =MBOX^  BSS    3               HP-IL Mailbox pointer
    244 2F7AC       =LOOPST BSS    1               HP-IL loop status
    245 2F7AD       =STATAR BSS    3               Statistical array name
    246 2F7B0       =TRACEM BSS    1               TRACE MODE (0,2,4,6)
    247 2F7B1       =DSPSET BSS    1               Display device set up on HPIL
    248             *
    249             *
    250             *
    251 2F7B2       =LOCKWD BSS    8*2             Password
    252             *
    253 2F7C2       =RESREG BSS    34              Result register
    254             *
    255             *
    256             *  ERR# through ERRL# are position dependent
    257             *
    258 2F7E4       =ERR#   BSS    4               Execution Error Number
    259 2F7E8       =CURRL  BSS    4               Current Line# Referenced
    260 2F7EC       =ERRL#  BSS    4               Execution Error Line#
    261             *
    262             *  Snapshot Buffer and Return Stack Save Buffer
    263             *
    264 2F7F0       =SNAPBF BSS    16+16+5+5+5     Snapshot Buffer
    265 2F81F       =RSTKBp BSS    1               Return Stack Save Buffer Pointer
    266 2F820       =RSTKBF BSS    16*5            Return Stack Save Buffer
    267             *
    268             *
    269             *
    270 2F870       =MLFFLG BSS    1               Multi-Line Function FLag
    271 2F871       =LBLIN#
    272 2F871       =STMTRO                        Statement scratch RAM
    273 2F871       =S-RO-0 BSS    5
    274 2F876       =S-RO-1 BSS    5
    275 2F87B       =S-RO-2 BSS    5
```

```
276 2F880       =S-R0-3 BSS    1
277             *
278 2F881       =STMTR1
279 2F881       =S-R1-0 BSS    5
280 2F886       =S-R1-1 BSS    5
281 2F88B       =S-R1-2 BSS    5
282 2F890       =S-R1-3 BSS    1
283             *
284 2F891       =STMTD0 BSS    5
285 2F896       =STMTD1 BSS    5
286             *
287 2F89B       =FUNCR0                        Function scratch RAM
288 2F89B       =F-R0-0 BSS    5
289 2F8A0       =F-R0-1 BSS    5
290 2F8A5       =F-R0-2 BSS    5
291 2F8AA       =F-R0-3 BSS    1
292             *
293 2F8AB       =FUNCR1
294 2F8AB       =F-R1-0 BSS    5
295 2F8B0       =F-R1-1 BSS    5
296 2F8B5       =F-R1-2 BSS    5
297 2F8BA       =F-R1-3 BSS    1
298             *
299 2F8BB       =FUNCD0 BSS    5
300 2F8C0       =FUNCD1 BSS    5
301             *
302             *  TRANSFORM Scratch RAM
303             *
304 2F8C5       =TRFMBF BSS    60              Used by TRANSFORM command
305             *
306             *
307 2F901       =SCRTCH                        Scratch RAM
308 2F901       =SCRST0 BSS    4*16            Scratch stack (Mantissas & signs)
309 2F941       =SCREX0 BSS    5               Scratch stack exponent 0
310 2F946       =SCROLT BSS    2               Character scroll timer   ---
311 2F948       =DELAYT BSS    2               Display timeout value      |
312 2F94A       =NEEDSC BSS    1               Scroll mode needed         | 11
313 2F94B       =PRMCNT BSS    2               CALL parameter count       |
314 2F94D       =DPOS   BSS    2               Current DISP column        |
315 2F94F       =DWIDTH BSS    2               DISP width               ---
316 2F951       =SCREX1 BSS    5               Scratch stack exponent 1
317 2F956       =PPOS   BSS    2               Current PRINT column     ---
318 2F958       =PWIDTH BSS    2               PRINT width                | 11
319 2F95A       =EOLLEN BSS    1               Length of ENDLINE string   |
320 2F95B       =EOLSTR BSS    2*3             ENDLINE string (3 chars max-
321 2F961       =SCREX2 BSS    5               Scratch stack exponent 2
322 2F966       =SCRPTR BSS    1               Scratch stack pointer    ---
323 2F967       =DEFADR BSS    8               Key definition info        | 11
324 2F96F       =CHN#SV BSS    2               Channel # save           ---
325 2F971       =SCREX3 BSS    5               Scratch stack exponent 3
326             *
327             *
328 2F976       =MAXCMD BSS    1               Number of command stack entries
329             *
330 2F977       =CSPEED BSS    5               Clock speed (Hz/16)
```

```
331             *
332             *
333             * The following 10 nibbles are used by HP-IL ROM
334             *
335 2F97C       =ERRLCH BSS    1              Error latch
336 2F97D       =TERCHR BSS    2              Terminating character for ENTER
337 2F97F       =HPSCRH BSS    7              HP-IL Reserved.
338                                              (INTPND,ICAUSE,IMASK,LSTDDC)
339             *
340             * The following RAM has been set aside for allocation at
341             *    a later time.  It is not used by the mainframe except
342             *    that is zeroed at cold start.
343             *
344 2F986       =RESERV BSS    48*2           Reserved Memory.
345             *
346             *      Configuration table and other I/O buffers
347             *
348 2F9E6       =CONFST
349 2F9E6               END
```

```
=ACTIVE  Abs  193960 #2F5A8 -   123
=ALRM1   Abs  194329 #2F719 -   222
=ALRM2   Abs  194341 #2F725 -   223
=ALRM3   Abs  194353 #2F731 -   224
=ALRM4   Abs  194365 #2F73D -   225
=ALRM5   Abs  194377 #2F749 -   226
=ALRM6   Abs  194389 #2F755 -   227
=ANN1.5  Abs  188673 #2E101 -    19
=ANNAD1  Abs  188672 #2E100 -    18
=ANNAD2  Abs  188674 #2E102 -    20
=ANNAD3  Abs  189260 #2E34C -    31
=ANNAD4  Abs  189262 #2E34E -    32
=ATNDIS  Abs  193601 #2F441 -    50
=ATNFLG  Abs  193602 #2F442 -    52
=AUTINC  Abs  194251 #2F6CB -   161
=AVMEME  Abs  193945 #2F599 -   115
=AVMEMS  Abs  193940 #2F594 -   110
=CALSTK  Abs  193965 #2F5AD -   124
=CHN#SV  Abs  194927 #2F96F -   324
=CHNLST  Abs  193982 #2F5BE -   130
=CLCBFR  Abs  193910 #2F576 -    98
=CLCSTK  Abs  193925 #2F585 -   101
=CMDPTR  Abs  194260 #2F6D4 -   163
=CMOSTV  Abs    5775 #0168F -    47
=CMOSTW  Abs  193592 #2F438 -    48
=CNTADR  Abs  194174 #2F67E -   141
=CONFST  Abs  195046 #2F9E6 -   348
=CSPEED  Abs  194935 #2F977 -   330
=CURREN  Abs  193900 #2F56C -    94
=CURRL   Abs  194536 #2F7E8 -   259
=CURRST  Abs  193885 #2F55D -    91
=CURSOR  Abs  193662 #2F47E -    81
=DATPTR  Abs  194194 #2F692 -   145
=DCONTR  Abs  189438 #2E3FE -    35
=DD1CTL  Abs  189439 #2E3FF -    36
=DD1END  Abs  189260 #2E34C -    30
=DD1ST   Abs  189184 #2E300 -    29
=DD2CTL  Abs  189183 #2E2FF -    28
=DD2END  Abs  189024 #2E260 -    26
=DD2ST   Abs  188928 #2E200 -    25
=DD3CTL  Abs  188927 #2E1FF -    24
=DD3END  Abs  188768 #2E160 -    22
=DD3ST   Abs  188676 #2E104 -    21
=DEFADR  Abs  194919 #2F967 -   323
=DELAYT  Abs  194888 #2F948 -   311
=DISINT  Abs  193648 #2F470 -    71
=DPOS    Abs  194893 #2F94D -   314
=DSPBFE  Abs  193856 #2F540 -    83
=DSPBFS  Abs  193664 #2F480 -    82
=DSPCHX  Abs  194164 #2F674 -   139
=DSPDGT  Abs  194269 #2F6DD -   206
=DSPFMT  Abs  194268 #2F6DC -   205
=DSPMSK  Abs  193856 #2F540 -    84
=DSPSET  Abs  194481 #2F7B1 -   247
=DSPSTA  Abs  193653 #2F475 -    78
```

```
=DVZNIB  Abs  194300 #2F6FC -   211
=DWIDTH  Abs  194895 #2F94F -   315
=EOLLEN  Abs  194906 #2F95A -   319
=EOLSTR  Abs  194907 #2F95B -   320
=ERR#    Abs  194532 #2F7E4 -   258
=ERRADR  Abs  194184 #2F688 -   143
=ERRL#   Abs  194540 #2F7EC -   260
=ERRLCH  Abs  194940 #2F97C -   335
=ERRSUB  Abs  194179 #2F683 -   142
=ESCSTA  Abs  193659 #2F47B -    79
=F-R0-0  Abs  194715 #2F89B -   288
=F-R0-1  Abs  194720 #2F8A0 -   289
=F-R0-2  Abs  194725 #2F8A5 -   290
=F-R0-3  Abs  194730 #2F8AA -   291
=F-R1-0  Abs  194731 #2F8AB -   294
=F-R1-1  Abs  194736 #2F8B0 -   295
=F-R1-2  Abs  194741 #2F8B5 -   296
=F-R1-3  Abs  194746 #2F8BA -   297
=FIRSTC  Abs  193660 #2F47C -    80
=FLGREG  Abs  194281 #2F6E9 -   188   205   206
=FORSTK  Abs  193950 #2F59E -   120
=FUNCD0  Abs  194747 #2F8BB -   299
=FUNCD1  Abs  194752 #2F8C0 -   300
=FUNCR0  Abs  194715 #2F89B -   287
=FUNCR1  Abs  194731 #2F8AB -   293
=GSBSTK  Abs  193955 #2F5A3 -   122
=HPSCRH  Abs  194943 #2F97F -   337
=INADDR  Abs  194260 #2F6D4 -   164
=INBS    Abs  194246 #2F6C6 -   160
=INTA    Abs  193552 #2F410 -    41
=INTB    Abs  193568 #2F420 -    42
=INTM    Abs  193584 #2F430 -    43
=INTR4   Abs  193536 #2F400 -    40
=INXNIB  Abs  194297 #2F6F9 -   208
=IOBFEN  Abs  193910 #2F576 -    97
=IOBFST  Abs  193905 #2F571 -    95
=IS-DSP  Abs  194445 #2F78D -   238
=IS-INP  Abs  194459 #2F79B -   240
=IS-PLT  Abs  194466 #2F7A2 -   241
=IS-PRT  Abs  194452 #2F794 -   239
=IS-TBL  Abs  194445 #2F78D -   237
=IVLNIB  Abs  194301 #2F6FD -   212
=KCOL0   Abs  193647 #2F46F -    69
=KCOL1   Abs  193646 #2F46E -    68
=KCOL2   Abs  193645 #2F46D -    67
=KCOL3   Abs  193644 #2F46C -    66
=KCOL4   Abs  193643 #2F46B -    65
=KCOL5   Abs  193642 #2F46A -    64
=KCOL6   Abs  193641 #2F469 -    63
=KCOL7   Abs  193640 #2F468 -    62
=KCOL8   Abs  193639 #2F467 -    61
=KCOL9   Abs  193638 #2F466 -    60
=KCOLA   Abs  193637 #2F465 -    59
=KCOLB   Abs  193636 #2F464 -    58
=KCOLC   Abs  193635 #2F463 -    57
```

```
=KCOLD   Abs  193634 #2F462 -    56
=KEYBUF  Abs  193604 #2F444 -    54
=KEYPTR  Abs  193603 #2F443 -    53
=KEYSAV  Abs  193634 #2F462 -    55
=LBLINM  Abs  194673 #2F871 -   271
=LDCSPC  Abs  194241 #2F6C1 -   158
=LEXPTR  Abs  194255 #2F6CF -   162
=LOCKWD  Abs  194482 #2F7B2 -   251
=LOOPST  Abs  194476 #2F7AC -   244
=MAINEN  Abs  193905 #2F571 -    96
=MAINST  Abs  193880 #2F558 -    88
=MAXCMD  Abs  194934 #2F976 -   328
=MBOX^   Abs  194473 #2F7A9 -   243
=MLFFLG  Abs  194672 #2F870 -   270
=MTHSTK  Abs  193945 #2F599 -   114
=NEEDSC  Abs  194890 #2F94A -   312
=NXTIRQ  Abs  194317 #2F70D -   221
=OFFFLG  Abs  193602 #2F442 -    51
=ONINTR  Abs  194189 #2F68D -   144
=OUTBS   Abs  193935 #2F58F -   108
=OVFNIB  Abs  194299 #2F6FB -   210
=PCADDR  Abs  194169 #2F679 -   140
=PNDALM  Abs  194401 #2F761 -   228
=PPOS    Abs  194902 #2F956 -   317
=PRGMEN  Abs  193895 #2F567 -    93
=PRGMST  Abs  193890 #2F562 -    92
=PRMCNT  Abs  194891 #2F94B -   313
=PRMPTR  Abs  193975 #2F5B7 -   129
=PWIDTH  Abs  194904 #2F958 -   318
=RAMEND  Abs  193970 #2F5B2 -   125
=RAWBFR  Abs  193920 #2F580 -   100
=RESERV  Abs  194950 #2F986 -   344
=RESREG  Abs  194498 #2F7C2 -   253
=RFNBFR  Abs  193915 #2F57B -    99
=RNSEED  Abs  194302 #2F6FE -   216
=ROWDVR  Abs  189264 #2E350 -    33
=RSTKBF  Abs  194592 #2F820 -   266
=RSTKBp  Abs  194591 #2F81F -   265
=S-R0-0  Abs  194673 #2F871 -   273
=S-R0-1  Abs  194678 #2F876 -   274
=S-R0-2  Abs  194683 #2F87B -   275
=S-R0-3  Abs  194688 #2F880 -   276
=S-R1-0  Abs  194689 #2F881 -   279
=S-R1-1  Abs  194694 #2F886 -   280
=S-R1-2  Abs  194699 #2F88B -   281
=S-R1-3  Abs  194704 #2F890 -   282
=SAVSTK  Abs  193950 #2F59E -   118
=SCREX0  Abs  194881 #2F941 -   309
=SCREX1  Abs  194897 #2F951 -   316
=SCREX2  Abs  194913 #2F961 -   321
=SCREX3  Abs  194929 #2F971 -   325
=SCROLT  Abs  194886 #2F946 -   310
=SCRPTR  Abs  194918 #2F966 -   322
=SCRST0  Abs  194817 #2F901 -   308
=SCRTCH  Abs  194817 #2F901 -   307
```

```
=SNAPBF  Abs  194544 #2F7F0 -  264
=STATAR  Abs  194477 #2F7AD -  245
=STMTD0  Abs  194705 #2F891 -  284
=STMTD1  Abs  194710 #2F896 -  285
=STMTR0  Abs  194673 #2F871 -  272
=STMTR1  Abs  194689 #2F881 -  278
=STSAVE  Abs  194238 #2F6BE -  157
=SYSEN   Abs  193930 #2F58A -  104
=SYSFLG  Abs  194265 #2F6D9 -  171
=TASTK   Abs  193945 #2F599 -  113
=TERCHR  Abs  194941 #2F97D -  336
=TFORN   Abs  193950 #2F59E -  119
=TGSBS   Abs  193955 #2F5A3 -  121
=TIMAF   Abs  194439 #2F787 -  235
=TIMER1  Abs  189432 #2E3F8 -   34
=TIMER2  Abs  189176 #2E2F8 -   27
=TIMER3  Abs  188920 #2E1F8 -   23
=TIMLAF  Abs  194427 #2F77B -  234
=TIMLST  Abs  194415 #2F76F -  233
=TIMOFS  Abs  194403 #2F763 -  232
=TMRAD1  Abs  194199 #2F697 -  146
=TMRAD2  Abs  194204 #2F69C -  147
=TMRAD3  Abs  194209 #2F6A1 -  148
=TMRIN1  Abs  194214 #2F6A6 -  153
=TMRIN2  Abs  194222 #2F6AE -  154
=TMRIN3  Abs  194230 #2F6B6 -  155
=TRACEM  Abs  194480 #2F7B0 -  246
=TRFMBF  Abs  194757 #2F8C5 -  304
=TRPREG  Abs  194297 #2F6F9 -  207
=UNFNIB  Abs  194298 #2F6FA -  209
=UPD1EN  Abs  193945 #2F599 -  111
=UPD1ST  Abs  193885 #2F55D -   90
=UPD2EN  Abs  194214 #2F6A6 -  149
=UPD2ST  Abs  194164 #2F674 -  138
=VECTOR  Abs  193596 #2F43C -   49
=WINDLN  Abs  193651 #2F473 -   77
=WINDST  Abs  193649 #2F471 -   76
```

Input Parameters

  Source file name is SB&RAM::MS

  Listing file name is SB/RAM:TI:ML::-1

  Object file name is SB%RAM:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

```
1           *    TTTTT  III   &     EEEEE   QQQQ   U   U
2           *      T     I   & &    E      Q    Q  U   U
3           *      T     I   & &    E      Q    Q  U   U
4           *      T     I    &     EEEE   Q    Q  U   U
5           *      T     I   & & &  E      Q  Q Q  U   U
6           *      T     I   &  &   E      Q   Q   U   U
7           *      T    III   && &  EEEEE   QQQ Q   UUU
8           *
9                    TITLE  Equate File <831228.1851>
10                   RDSYMB SB%TAB::MS
11                   RDSYMB JP%TAB::MS
12
13
14          ************************************************************************
15          **                                                                    **
16          **              SYSTEM GLOBAL SYMBOL DEFINITIONS                      **
17          **                                                                    **
18          ************************************************************************
19
20
```

```
                STITLE CALL Stack Save Blocks
**============================================================
**
**            CALL Stack Save Blocks
**
**============================================================

**
*
* Mainframe Save Block Pointers
*
=caCURS EQU   6              Offset to saved CURRST
=caPRMS EQU   (caCURS)+5             //       PRGMST
=caPRME EQU   (caPRMS)+5             //       PRGMEN
=caCURE EQU   (caPRME)+5             //       CURREN
=caPCAD EQU   (caCURE)+5             //       PCADDR
=caCNTA EQU   (caPCAD)+5             //       CNTADR
=caERRS EQU   (caCNTA)+5             //       ERRSUB
=caERRA EQU   (caERRS)+5             //       ERRADR
=caINTR EQU   (caERRA)+5             //       ONINTR
=caDATP EQU   (caINTR)+5             //       DATPTR
=caFSTK EQU   (caDATP)+5             //       FORSTK offset
=caGSTK EQU   (caFSTK)+5             //       GSBSTK offset
=caACTV EQU   (caGSTK)+5             //       ACTIVE offset
=caCALS EQU   (caACTV)+5             //       CALSTK offset
=caPMCN EQU   (caCALS)+5             //       PRMCNT offset
=caPMTR EQU   (caPMCN)+2             //       PRMPTR offset
=caRTNT EQU   (caPMTR)+5      CALL return type
=caLENG EQU   (caRTNT)+1      CALL save block length

**
*
* User-defined Function Save Block Pointers
*
=ufRTNA EQU   6             Offset to function return addr
=ufPC   EQU   (ufRTNA)+5    Offset to saved PC
=ufCNTA EQU   (ufPC)+5      Offset to saved CONTADR
=ufSTM0 EQU   (ufCNTA)+5            //       STMT0
=ufRSTK EQU   (ufSTM0)+5            //       3 RSTKs
=ufSTM1 EQU   (ufRSTK)+15           //       STMT1
=ufSR-0 EQU   (ufSTM1)+5            //       S-R0-0
=ufSR-1 EQU   (ufSR-0)+16           //       S-R1-0
=ufMSTK EQU   (ufSR-1)+16           //       MTHSTK
=ufFSTK EQU   (ufMSTK)+5            //       FORSTK
=ufGSTK EQU   (ufFSTK)+5            //       GSBSTK
=ufPMCN EQU   (ufGSTK)+5            //       PRMCNT
=ufPMTR EQU   (ufPMCN)+2            //       PRMPTR
=ufSTSV EQU   (ufPMTR)+5            //       STSAVE
=ufCHN# EQU   (ufSTSV)+3            //       CHN#SV
=ufRTNT EQU   (ufCHN#)+2     Function return type
=ufLENG EQU   (ufRTNT)+1     Save block length
```

```
                STITLE Complex Poll Sub-subfunctions
**============================================================
**
**        Complex Poll Sub-subfunction Definitions
**
**      Note:  These sub-subfunctions are handled by the
**             pCOMPLX poll.
**
**============================================================
=cRCL   EQU    #67
=cC->C  EQU    #68
=cR->C  EQU    #69
```

```
                STITLE Device Types
          **============================================================
          *
          *                          Device Types
          *
          **============================================================
          =dMAIN  EQU   0             MAIN
          =dIRAM  EQU   1             IRAM Device
          =dPORT  EQU   1             PORT Device
          =dCARD  EQU   7             CARD Device
          =dPCRD  EQU   7             PCRD Device
```

```
                STITLE File Header Equates
**============================================================
**
**                File Header Equates
**
** Detail:
**
**   File name:
**      8 characters, blank filled
**
**   File type:
**      LIF File Type Number of the file.  This field ALWAYS
**      represents the unprotected form of the file type
**
**   Flags:
**      a) Lower nibble of the flags:
**         This nibble is called the protection nibble.
**         Bit:    3    2    1    0
**              --------------------      PV - Private
**              | Un | Un | PV | SE |      SE - Secure
**              --------------------      Un - Undefined
**
**      b) Higher nib of the flags:
**         File copy code from file type table.  Its value
**         determines how the file may be accessed:
**
**         Value              Meaning
**         -----    ----------------------------------------
**           0      Normal mainframe file structure;
**                  File can be copied into or out of
**                  HP-71 without aid from LEX file;
**                  Implementation Field contains
**                  file length on external copy,
**                  but is not present after file
**                  header when file is in memory
**
**           1      DATA file structure; file can be
**                  copied into or out of HP-71 with
**                  no aid from LEX file; on external
**                  copy, the Implementation Field
**                  contains number of records and
**                  record length, and it is present
**                  immediately after file header
**                  when file is in memory
**
**           2      SDATA file structure; file can be
**                  copied into or out of HP-71 with
**                  no aid from LEX file; on external
**                  copy, the Implementation Field
**                  contains number of records, but
**                  is not present after file header
**                  when file is in memory
**
**           4      TEXT file structure; file can be
**                  copied into or out of HP-71 with
```

```
150             **                no aid from LEX file; on external
151             **                copy, the Implementation Field is
152             **                zero, and it is not present after
153             **                file header when file is in
154             **                memory
155             **
156             **          8     Special copy routine is required
157             **                to copy file to or from HP-71;
158             **                system will issue pWCRD8 poll;
159             **                Implementation Field is present
160             **                after file header when file is
161             **                in memory
162             **
163             **    Time Field:
164             **       Time of creation of file
165             **
166             **    Date:
167             **       Date of creation of file
168             **
169             **    File Chain Length:
170             **       Five-nibble relative address of the next file in the
171             **       chain.
172             **
173             **    Implementation Field:
174             **       When present, this field is 8 nibbles long.  It
175             **       corresponds directly with the Implementation Field
176             **       in the file's directory entry when written to LIF
177             **       external media.
178             **
179             *
180             =oFNAMh EQU    #00000          Offset Filename in header
181             =oFTYPh EQU    #00010          Offset File Type in header
182             =oFLAGh EQU    #00014          Offset to Flags in header
183             =oTIMEh EQU    #00016          Offset Time in header
184             =oDATEh EQU    #0001A          Offset Date in header
185             =oFLENh EQU    #00020          Offset File Length in header
186             =oIMPLh EQU    #00025          Offset to Implementation Field
187             =oSUBLn EQU    #00025          Offset to sub-program chain head
188             =oFLSTr EQU    #00031          Offset to start of BASIC file
189             *
190             =oFT-FL EQU    (oFLENh)-(oFTYPh)
191             *
192             =lFNAMh EQU    #00010          Length Filename in header
193             =lFTYPh EQU    #00004          Length File Type in header
194             =lFLAGh EQU    #00002          Length Flags in header
195             =lTIMEh EQU    #00004          Length Time in header
196             =lDATEh EQU    #00006          Length Date in header
197             =lFLENh EQU    #00005          Length File Length in header
198             *
199             =oKYsod EQU    #00005          Offset to data - KEY file
200             =oBNsod EQU    #00011          Offset to data - BINARY file
201             =oBSsod EQU    #00011          Offset to data - BASIC file
202             =oDAsod EQU    #0000D          Offset to data - DATA file
203             =oLXsod EQU    #00005          Offset to data - LEX file
204             =oTXsod EQU    #00005          Offset to data - TEXT (LIF1) file
```

```
205             =o41sod EQU    #00005          Offset to data - SDATA file
206             =lEOL   EQU    #2              Length EOL
207
```

```
208                     $TITLE File Information Buffer Equates
209             **=============================================================
210             **
211             **          File Information Buffer (FIB) Equates
212             **
213             **  The FIB contains the following information:
214             **
215             **  1. FIB number (2 nibs) - If 00, end of FIB
216             **
217             **  2. File buffer number (3 nibs) -
218             **       If (and only if) the file is on an external device,
219             **       it has an I/O buffer associated with it, and this
220             **       field contains the I/O buffer number of the file.  If
221             **       this field is 000, the file is in memory and there is
222             **       no I/O buffer associated with it.
223             **
224             **  3. File type (4 nibs) - LIF file type of the file
225             **       This is the unprotected file type number of the file.
226             **
227             **  4. File protection nibble (1 nibs) - Copied from the low
228             **       nibble of the file flags field in the file header
229             **       when the file is opened.  Subsequent SECURE or
230             **       PRIVATE commands do not alter its value.  Original
231             **       value derives from the file type table.
232             **
233             **  5. File copy code (1 nib) - This is the same nibble as in
234             **       the file header.
235             **
236             **  6. Access code (1 nib) -
237             **       When set to 1, this field indicates that the contents
238             **       of the file have been modified since the field was
239             **       last cleared.  For external files, it is cleared each
240             **       time a new sector is read into the I/O buffer.  It is
241             **       set when the contents of the I/O buffer are modified
242             **       so that the I/O buffer can be flushed properly during
243             **       subsequent operations.
244             **
245             **  7. Device type (1 nib) - This nibble indicates where the
246             **      file is located:
247             **       0 - Mainframe          1 - Independent RAM
248             **       2 - ROM                8 - HP-IL device
249             **
250             **  8. File begin (6 nibs) -
251             **      For file in Memory:
252             **        Nibs     Field
253             **        -----    ----------------------------------------
254             **        4-0      Absolute address of file header start
255             **        5        Unused
256             **      For file on external device:
257             **        Nibs     Field
258             **        ------   ----------------------------------------
259             **        0        Entry number in the directory record
260             **        4-1      Sector number of the directory record
261             **
262             **  9. Subheader length (2 nibs)
```

```
263           **       This length is the number of bytes of the subheader.
264           **       It is computed as follows:
265           **       a) Take the data offset (in nibs) from the file type
266           **          table and subtract 5 nibbles from it.
267           **       b) If the file type is 1 (fixed length data format)
268           **          subtract another 8 nibbles (implementation field).
269           **       c) Divide by 2 to convert nibbles into bytes.
270           **
271           **       Copy code 1: (data offset-13) / 2
272           **       Copy codes 0,2,4: (data offset -5) / 2
273           **       Copy code 8: Set by poll handler when file opened.
274           **
275           ** 10. File data start (11 nibs) -
276           **       Absolute address of the start of data of the file.
277           **
278           **       For file in RAM/ROM:
279           **          Nibs    Field
280           **          ----    -----
281           **          4-0     Abs addr of data start
282           **                  (skip over the subheader)
283           **          6-5     Hex F0
284           **          8-7     Port address: Port #, Extender #
285           **         10-9     Unused
286           **
287           **       For file on an external (HP-IL) device:
288           **          Nibs    Field
289           **          ----    --------------------------------
290           **          3-0     Record address of the first record of the
291           **                  file.  If file has a subheader, it starts
292           **                  at byte 00 of this record and is immedia-
293           **                  tely followed by data.  Otherwise, the
294           **                  data starts at byte 00 of this record.
295           **          6-4     Device address
296           **          9-7     Assign code
297           **          10      Assign type
298           **
299           **  11. Number of Records (4 nibs) -
300           **        For copy code 1 (DATA) files, this is the number of
301           **        records in the file.
302           **
303           **  12. Record length (4 nibs) -
304           **        For copy code 1 (DATA) files, this is the record
305           **        length in bytes.
306           **
307           **  13. Current position (6 nibs) -
308           **        This is the current file pointer. It is the offset
309           **        from the file start reference point.  Due to the
310           **        sector boundary constraints of HPIL I/O, the file
311           **        start reference point for external files differs
312           **        slightly from that of internal files.
313           **
314           **          Nibs     Field
315           **          -----    ----------------------------------------
316           **          6-5      Not used
317           **          4-0      Offset from file start reference point in
```

```
318           **                  nibbles.
319           **                  For internal files:
320           **                   Offset from start of data (that is, sub-
321           **                   header not included)
322           **                  For external files:
323           **                   Offset from start of subheader
324           **
325           **  14. File data length (6 nibs) -
326           **        This is the file data length not including the
327           **        subheader.
328           **          Nibs     Field
329           **          -----    ----------------------------------------
330           **          5-0      File data length in nibbles.
331           **
332           **  15. Remaining length in current record (5 nibs) -
333           **        Used by PRINT# and READ# to keep track of the number
334           **        of bytes remaining in the current record for copy
335           **        code 1 (DATA) files.
336           **
337           **  16. Device size (6 nibs)
338           **        This field applies only for files in an external
339           **        memory device (HP-IL)
340           **          Nibs     Field
341           **          -----    ----------------------------------------
342           **          5-0      Number of sectors allocated to this file.
343           **
344           **=============================================================
345           *
346           *
347           =lFIL#b EQU    #00002          Length File# in FIB
348           =lFBF#b EQU    #00003          Length File buffer # in FIB
349           =lFTYPb EQU    #00004          Length File Type in FIB
350           =lPROTb EQU    #00001          Length Protection in FIB
351           =lCOPYb EQU    #00001          Length Copy code in FIB
352           =lACCSb EQU    #00001          Length Access code in FIB
353           =lDEVCb EQU    #00001          Length Device ID in FIB
354           =lFBEGb EQU    #00006          Length File header/Dir entry addr
355           =lSHLNb EQU    #00002          Length Subheader of the file
356           =lDBEGb EQU    #0000B          Length Data Start
357           =lREC#b EQU    #00004          Length Number of Records in FIB
358           =lRECLb EQU    #00004          Length Record Length in FIB
359           =lCPOSb EQU    #00006          Length Current Postion in file
360           =lDLENb EQU    #00006          Length File data Length in bytes
361           =lRLENb EQU    #00005          Length Remaining bytes in record
362           =lFSIZb EQU    #00006
363           =lFIB   EQU    #0003F          Length of File Information Buffer
364           *
365           =oFIL#b EQU    #00000              Offset File#
366           =oFBF#b EQU    (oFIL#b)+(lFIL#b)   Offset File buffer #
367           =oFTYPb EQU    (oFBF#b)+(lFBF#b)   Offset File Type
368           =oPROTb EQU    (oFTYPb)+(lFTYPb)   Offset Protection
369           =oCOPYb EQU    (oPROTb)+(lPROTb)   Offset copy code
370           =oACCSb EQU    (oCOPYb)+(lCOPYb)   Offset Access code
371           =oDEVCb EQU    (oACCSb)+(lACCSb)   Offset Device ID
372           =oFBEGb EQU    (oDEVCb)+(lDEVCb)   Offset File header/Dir entry
```

```
373             *                                       address
374             =oSHLNb EQU     (oFBEGb)+(1FBEGb)   Offset subheader length
375             =oDBEGb EQU     (oSHLNb)+(1SHLNb)   Offset Data Start
376             =oREC#b EQU     (oDBEGb)+(1DBEGb)   Offset # of records
377             =oRECLb EQU     (oREC#b)+(1REC#b)   Offset Record Length
378             =oCPOSb EQU     (oRECLb)+(1RECLb)   Offset Current Position
379             =oDLENb EQU     (oCPOSb)+(1CPOSb)   Offset File data Length
380             =oRLENb EQU     (oDLENb)+(1DLENb)   Offset Remaining Length
381             =oFSIZb EQU     (oRLENb)+(1RLENb)   Offset File size in sectors
```

```
382                     STITLE File Type Symbol Definitions
383             **============================================================
384             **
385             **                  File Type Symbol Definitions
386             **
387             **============================================================
388             =fBASIC EQU    #E214          BASIC File Type
389             =fASCII EQU    #0001          ASCII Type 1 File Type
390             =fKEY   EQU    #E20C          KEY File Type
391             =fLIF1  EQU    #0001          LIF1 File Type
392             =fTEXT  EQU    #0001          TEXT File Type
393             =fSDATA EQU    #E0D0          Stream Data File Type
394             =fDATA  EQU    #E0F0          Fixed Data File Type
395             =fBIN   EQU    #E204          Binary File Type
396             =fLEX   EQU    #E208          Lex File Type
397             *
398             * Record Header of Copy Code 1 (DATA) Files
399             *
400             =fEOF   EQU    #00FF          End of file mark
401             =fEOR   EQU    #00EF          End of record mark
402             =fAOS   EQU    #00DF          Entire string in the same record
403             =fSOS   EQU    #00CF          Start of string header
404             =fMOS   EQU    #007F          Middle of string header
405             =fEOS   EQU    #006F          End of string header
```

```
406                 STITLE LEX File Format Equates
407             **============================================================
408             **
409             **                    LEX File Format Equates
410             **
411             **
412             **   LEX File Format:
413             **                                              Size
414             **          Field                              (nibs)
415             **          -------------------------------    ------
416             **          Standard File Header                 49
417             **          Lex ID                                2
418             **          Token Range                           4
419             **          Lex Table Chain                       5
420             **          Speed Table nibble                    1
421             **          SPEED Table          (optional)      78  (3*26)
422             **          Speed Table nibble 2 (optional)       1
423             **          Text Table pointer                    4
424             **          Message Table pointer                 4
425             **          Poll Address pointer                  5
426             **          MAIN Table                            *
427             **          Text Table                            *
428             **          Message # Range                       4
429             **          Message Table                         *
430             **
431             **============================================================
432             =lLXID  EQU    2              Length of LEX ID
433             =lLXTKR EQU    4              Length of Lex Token Range
434             =lLXFAD EQU    5              Length of Lex Table Chain Field
435             =lSPDn  EQU    1              Length of Speed Table nibble
436             =lSPDn2 EQU    1              Length of Speed Table nibble 2
437             =lTEXTp EQU    4              Length of Text Table Offset field
438             =lMSGp  EQU    4              Length of Mesage Tbl Offset field
439             =lPOLLp EQU    5              Length of Poll Offset field
440             =lSPDTB EQU    3*26           Len of Speed Table=#nibs*#entries
441
442             **
443             *
444             * Offset to Main Table in LEX File:
445             *
446             *   Speed Table nibble + Speed Table + Speed Table nibble 2 +
447             *   Text Table Ptr + Msg Table Ptr + Poll Address Ptr
448             *
449             =oMAINT EQU (lSPDn)+(lSPDTB)+(lSPDn2)+(lTEXTp)+(lMSGp)+(lPOLLp)
450
451             **
452             *
453             * Offset to Speed Nibble 2 =
454             *   Poll Addr Ptr + Message Table Ptr + Text Table Ptr +
455             *   + length Speed nib 2
456             *
457             * Offset to Text Table = position of Speed nibble 2, then
458             *                        read 5 nibs
459             *
460             =oSPDn2 EQU    (lPOLLp)+(lMSGp)+(lTEXTp)+(lSPDn2)
```

```
461
462             **
463             *
464             * Offset to Speed Table from Main Table Start =
465             *   # nibbles/table entry * ASCII "Z" - 1
466             *
467             =oSPDTB EQU    3*(\Z\+1)
468
469             **
470             *
471             * Offset to Message Table pointer from Main Table Start =
472             *   Poll Address ptr + Message Table Pointer
473             *
474             =oMSGPT EQU    (lPOLLp)+(lMSGp)
475
476             **
477             *
478             * LEX Buffer Equates
479             *
480             =lLXADR EQU    5               Length of LEX Main Table Address
481             =lLXENT EQU    (lLXID)+(lLXTKR)+(lLXADR) Leng LEX Buffer Entry
482
```

```
                STITLE Poll Process Numbers
**============================================================
**
**                    Poll Process Numbers
**
**         Those marked with "*" are processed by HPIL.
**
**============================================================
*
* VER$
*
=pVER$  EQU    #00      *     VER$ poll
*
* File Spec
*
=pDEVCp EQU    #01      *     Device Parse
=pFILDC EQU    #02      *     File Spec Decompile
=pFILXQ EQU    #03            File Execute:
*                               allow dedicated device
=pFSPCp EQU    #04      *     File Spec Parse
=pFSPCx EQU    #05      *     File Spec Execute
*
* External device
*
=pCAT   EQU    #06      *     CAT on non-mainframe device
=pCAT$  EQU    #07      *     CAT$ of non-mainframe file
=pCOPYx EQU    #08      *     COPY execute: unknown Device|>8
=pCREAT EQU    #09      *     Create file in external device
=pDIDST EQU    #0A      *     Device ID store in RAM @ D1
=pFPROT EQU    #0B      *     SECURE/UNSECURE/PRIVATE
=pLIST  EQU    #0C            LIST of non-mainframe file
=pMERGE EQU    #0D            MERGE file dealing w/funny device
=pPRTCL EQU    #0E      *     Print class
=pPRTIS EQU    #0F      *     Printer IS
=pPURGE EQU    #10      *     PURGE on non-mainframe device
=pRNAME EQU    #11      *     RENAME on non-mainframe device
=pENTER EQU    #12      *     Enter data from HP-IL
*              #13      *     Reserved for HPIL
*              #14      *     Reserved for HPIL
*              #15      *     Reserved for HPIL
*              #16      *     Reserved for HPIL

*
* File I/O
*
=pFINDF EQU    #17      *     Find file
=pRDCBF EQU    #18      *     Read current recrd to file buffer
=pRDNBF EQU    #19      *     Write buffer out, read next recrd
=pWRCBF EQU    #1A      *     Write buffer to current recrd
*
* Keyboard
*
=pKYDF  EQU    #1B      *     Build key defn in KEYRD
=pWTKY  EQU    #1C            Waiting for key in KEYRD
*
```

```
538             * Image
539             *
540             =pIMXQT EQU    #1D      *     IMAGE execution starts
541             =pIMCHR EQU    #1E            Unrecognized IMAGE char in parse
542             =pIMXCH EQU    #1F            Unrecognized IMAGE symbol in exec
543             =pIMbck EQU    #20            IMAGE: bckwd search processing
544             =pIMcpi EQU    #21            IMAGE: cmplx field initialization
545             =pIMcpw EQU    #22            IMAGE: work on complex number
546             *
547             * Copy Code = 8
548             *
549             =pCRT=8 EQU    #23            Create non-Titan type file
550             =pWCRD8 EQU    #24            Write card, copycode=8
551             *
552             * READ#/PRINT#
553             *
554             =pEOFIL EQU    #25            End of file in READ #/PRINT #
555             =pPRIN# EQU    #26            PRINT # on non-Titan type file
556             =pREAD# EQU    #27            READ # on non-Titan type file
557             =pSREC# EQU    #28            RESTORE # on non-Titan type file
558             *
559             * Unknown file type
560             *
561             =pCURSR EQU    #29            Cursor Up/Down for non-BASIC file
562             =pDATLN EQU    #2A            Return file data length on
563             *                               non-HP71 file
564             =pEDIT  EQU    #2B            EDIT with non-BASIC file type
565             =pFASCH EQU    #2C            Search for filetype by mnemonic
566             =pFTYPE EQU    #2D            File type
567             =pLIST2 EQU    #2E            LIST non-BASIC/non-KEY file
568             =pMRGE2 EQU    #2F            MERGE non-BASIC/non-KEY file
569             =pRUNft EQU    #30            RUN with unknown File Type
570             =pRUNnB EQU    #31            RUN non-BASIC file
571             *
572             * Unknown memory type
573             *
574             =pPRGPR EQU    #32            PURGE of non-RAM file
575             *
576             * Card Reader
577             *
578             =pCRDAB EQU    #33            Abort card read poll
579             =pRCRD  EQU    #34            Read card poll
580             =pWCRD  EQU    #35            Write card poll
581             *
582             * CALL
583             *
584             =pCALRS EQU    #36            Restore information from CALL stk
585             =pCALSV EQU    #37            Save information on CALL stack
586             *
587             * Miscellaneous
588             *
589             =pCMPLX EQU    #38            Complex math
590             =pREN   EQU    #39            Renumber a XWORD stmt with line #
591             =pRTNTp EQU    #3A            Return Type unknown
592             =pTIMR# EQU    #3B            Timer # > 3 in ON TIMER/OFF TIMER
```

```
593           =pTRFMx EQU   #3C           Supply Transform Handler Address
594           =pFNIN  EQU   #3D           Entering user-defined function
595           =pFNOUT EQU   #3E           Exiting user-defined function
596           *
597           * System
598           *
599           =pTRANS EQU   #EF           Poll to Translate a Message
600           =pTEST  EQU   #F0           Test poll for timing POLLs
601           =pMEM   EQU   #F1           Insufficient Memory
602           =pERROR EQU   #F2           Error message about to go out
603           =pWARN  EQU   #F3           Warning msg about to go out
604           =pPARSE EQU   #F4           Parse take-over poll - FAST Poll
605           =pBSCen EQU   #F5           Entering BASIC interpreter
606           =pBSCex EQU   #F6           Exiting BASIC interpreter
607           =pZERPG EQU   #F7     *     Zero addrs/RAM associated w/Prgm
608           =pExcpt EQU   #F8     *     Exception check after statement
609           =pSREQ  EQU   #F9     *     Service request (if SREQ<>0)
610           =pMNLP  EQU   #FA     *     Main Loop
611           =pCONFG EQU   #FB     *     Configuration
612           =pPWROF EQU   #FC     *     Power off
613           =pDSWKY EQU   #FD     *     Deep Sleep Wakeup -- key or not
614           =pDSWNK EQU   #FE     *     Deep Sleep Wakeup -- no key down
615           =pCLDST EQU   #FF     *     Cold start
```

```
616                     STITLE Print Class Statement Types
617             **============================================================
618             **
619             **                  Print Class Statement Types
620             **
621             **============================================================
622
623             =DISPt  EQU     0               DISP statement
624             =PRINTt EQU     1               PRINT statement
625
```

```
626                     STITLE Save Stack Offset Equates
627               **============================================================
628               **
629               **                  Save Stack Offset Equates
630               **
631               **============================================================
632
633               **
634               *
635               * COPY/TRANSFORM File Save Information Offsets
636               *
637               *    Detail:
638               *      SAVSTK-50  Destination Filename
639               *      SAVSTK-30  Destination Device Informaiton
640               *      SAVSTK-25  Source Filename
641               *      SAVSTK- 5  Source Device Information
642               *
643               =lFILSV EQU    50              Length of File Info Save area for
644               *                              COPY
645               =lFNAM8 EQU    16              First 8 chars of filename
646               =lFNAM+ EQU    4               Last 2 chars of filename
647               =lDEVC  EQU    5               Device Information
648
649               **
650               *
651               *            POLL Information Save Stack Offsets
652               *
653               *   Purpose:
654               *     The process of issuing a poll places the following data
655               *     on the SAVSTK, which may be referenced using the offsets
656               *     defined below.  Offsets are negative in direction (that
657               *     is, toward lower addresses) and relative to the current
658               *     value of the SAVSTK pointer.
659               *
660               *   Note:
661               *     RTN1p and POL# occupy the same position
662               *
663               =lBPOSp EQU    #5              Relative Buffer Position Length
664               =lRTN1p EQU    #5              Return Address 1  Length
665               =lPOL#p EQU    #5              Poll# Length
666               =lRTN2p EQU    #5              Return Address 2  Length
667               =lRTN3p EQU    #5              Return Address 3  Length
668               =lD0p   EQU    #5              D0
669               =lD1p   EQU    #5              D1
670               =lDp    EQU    #10             D
671               =lAp    EQU    #10             A
672               *
673               =oBPOSp EQU    #5              Buffer Position Offset
674               =oRTN1p EQU    #A              Return Address 1 Offset
675               =oPOL#p EQU    #A              Poll # Offset
676               =oRTN2p EQU    #F              Return Address 2 Offset
677               =oRTN3p EQU    #14             Return Address 3 Offset
678               =oD0p   EQU    #19             D0 Offset
679               =oD1p   EQU    #1E             D1 Offset
680               =oDp    EQU    #2E             D  Offset
```

```
681             =oAp    EQU    #3E            A  Offset
682             *
683             =lPOLSV EQU    oAp            POLL Save Info Length
684             =lPOLra EQU    6              POLL return address on GSBSTK
```

```
                STITLE Status Bit Definitions
              **============================================================
              *
              *                  Status Bit Definitions
              *
              **============================================================

              **
              *  Global Status Symbols (Almost Always Active)
              *
              =Except EQU    12              An exception has occurred
              =PgmRun EQU    13              Program is running
              =NoCont EQU    14              Don't continue running
              =Trace  EQU    15              Trace mode active

              **
              *  Miscellaneous Status Symbols
              *
              =sBYEx  EQU    0               BYE or OFF Execute
              =sERROR EQU    0               Execution ERROR occured
              =sEXTDV EQU    0               HPIL Device
              =sRETRN EQU    0               RETURN statement
              =sENDx  EQU    1               END/STOP Execute
              =sSSTdc EQU    1               Single Step Decompile
              =sUNDEF EQU    1               File Names Undefined
              =sCARD  EQU    2               CARD or PCRD (for MFDEVC)
              =sCURUP EQU    2               Cursor UP flag
              =sIRAM  EQU    2               IRAM Destination for COPY
              =sNoChn EQU    2               No File Chain in ROM/RAM
              =sSST   EQU    2               SST at End of Program
              =sCURBT EQU    3               Cursor BOTTOM flag
              =sDEST  EQU    3               Dest Exec in COPY
              =sGOSUB EQU    3               GOSUB statement
              =sCURUD EQU    4               NOT Cursor UP or DOWN
              =sONERR EQU    4               ON ERROR statement
              =sREADI EQU    4               Read File Information
              =sRUNBn EQU    4               RUN Binary file
              =sEXTGS EQU    5               External entry to GOSUB
              =sKEYS  EQU    5               COPY to KEYS
              =sMAINc EQU    5               MAINframe CHAIN
              =sONTMR EQU    6               ON TIMER statement
              =sRENAM EQU    6               RENAME execute
              =sSTAT  EQU    6               Statistical Operations
              =sARITH EQU    7               Arithmetic
              =sEOF   EQU    7               End of File (TRANSFORM)
              =sRUNDC EQU    7               RUN Decompile
              =sCARDC EQU    8               CARD Run
              =sPCRD  EQU    8               Private Card
              =sRENUM EQU    8               Renumber Parse flag
              =sRFILE EQU    8               RUN File specified in Parse
              =sCONTK EQU    9               CONT Key entered
              =sXWORD EQU    9               XWORD entry for GOTO+
              =sCONT  EQU    10              CONT statement
              =sRESTR EQU    10              RESTORE statement
              =sCHAIN EQU    11              CHAIN execute
```

```
740                     STITLE System Buffer IDs
741               **============================================================
742               **
743               **                          System Buffer IDs
744               **
745               **  Detail:
746               **    A System Buffer ID is a 3-nibble value.  If the high bit
747               **    is set, the buffer will be kept (not automatically purged
748               **    by the system) during configuration.
749               **
750               **============================================================
751
752               =bSTMT  EQU    #801           Statement Buffer ID
753               =bIEXKY EQU    #802           Immediate Execute Key Buffer
754               *
755               =bFIB   EQU    #803           File Information Buffer ID
756               *
757               =bASSGN EQU    #804           I/O buf ID for ASSIGN
758               =bFILE  EQU    #805           Temp. buf for file manipulation
759               =bSTAT  EQU    #806           Statistics
760               *
761               =bCARD  EQU    #807           CARD reader Buffer ID
762               =bSTART EQU    #808           STARTUP Buffer ID
763               =bECOMD EQU    #809           External COMmanD buffer
764               *
765               * HP-IL buffers
766               *
767               =bPILSV EQU    #80F           HPIL save area
768               =bPILAI EQU    #810           ASSIGNIO names
769               =bSTMXQ EQU    #811           HP-IL statement execution buffer
770               *
771               =bALTCH EQU    #BFB           Alternate Charset Buffer ID
772               =bCHARS EQU    #BFB           Alternate Charset Buffer ID
773               =LXBFID EQU    #BFC           Lex Buffer ID
774               =bLEX   EQU    #BFC           Lex Buffer ID
775               =ROMCID EQU    #BFE           ROM Configuration Table
776               =bROMTB EQU    #BFE           ROM Configuration Table
777               *
778               * Buffers #E00 - #FFF reserved for scratch buffers
779               *
780               =bSCRTC EQU    #E00           Start of SCRTCH buffers
```

```
                STITLE System Constants
**============================================================
**
**                    System Constants
**
**============================================================

=LEEWAY EQU    50+50+50+62    COPY save(50) + Command Stk(50)
*                              + bSTMT(50) + Poll(62)
=BASICs EQU    (LASTFN)+1     BASIC Command Start for Run Loop

**
* Hardware addresses
*
=CR     EQU    #2C000         Card Reader
```

```
                STITLE System Flag Definitions
**============================================================
**
**        System Flag Definitions (Range -1 to -64)
**
**  Note: Flag symbols should begin with "fl", rather than "s".
**
**============================================================

**
**        <<<< Test, Set, and Cleared by User >>>>
**

*
*> Nib boundary                                          Flag
*                                                        ----
=flQIET EQU   00-01          Quiet Mode                  (-1)
=flBEEP EQU   00-02          Beep On                     (-2)
=flCTON EQU   00-03          Continuous On               (-3)
=flINX  EQU   00-04          Inexact result              (-4)
*
*> Nib Boundary
*
=flUNF  EQU   00-05          Underflow                   (-5)
=flOVF  EQU   00-06          Overflow                    (-6)
=flDVZ  EQU   00-07          Divide by Zero              (-7)
=flIVL  EQU   00-08          Invalid operation           (-8)
*
*> Nib Boundary
*
=flUSER EQU   00-09          User Mode set               (-9)
=flRAD  EQU   00-10          RAD trig mode               (-10)
=flINFR EQU   00-11          Round to Infinity           (-11)
=flNEGR EQU   00-12          Negative Round              (-12)
*
*> Nib Boundary
*
=flFXEN EQU   00-13          FIX/ENG flag                (-13)
=flSCEN EQU   00-14          SCI/ENG flag                (-14)
=flLC   EQU   00-15          Lower Case enabled          (-15)
=flBASE EQU   00-16          Base Option   (high bit!)(-16)
*
*> Nib Boundary
*
=flDG0  EQU   00-17          Display digit bit 0         (-17)
=flDG1  EQU   00-18          Display digit bit 1         (-18)
=flDG2  EQU   00-19          Display digit bit 2         (-19)
=flDG3  EQU   00-20          Display digit bit 3         (-20)
*
*> Nib Boundary
*
=flPDWN EQU   00-21    HPIL: Don't pwr loop down autom(-21)
=flEXTD EQU   00-22    HPIL: Use extended addressing   (-22)
=flEOT  EQU   00-23    HPIL: Entry terminated by EOT   (-23)
=flNZ4  EQU   00-24          "                         (-24)
```

```
851             *
852             *> Nib Boundary
853             *
854             =f1BPLD EQU    00-25          Beep LOUD                 (-25)
855             =f1NOPR EQU    00-26          Don't Prompt              (-26)
856
857
858             **
859             **                  <<<< Test Only >>>>
860             **
861
862             *
863             *> Nib boundary                                          Flag
864             *                                                        ----
865             =f1MPI  EQU    00-41          Module has been pulled    (-41)
866             ***
867             *** NOTE:
868             ***   Value of f1NOFN must end in 9 or 1 (see message
869             ***   routines in TI&ERD)
870             ***
871             =f1NOFN EQU    00-42          No user-defined functions(-42)
872             =f1DORM EQU    00-43          Machine is dormant        (-43)
873             =f1RTN  EQU    00-44          Always Return from MEMERR(-44)
874             *
875             *> Nib Boundary
876             *
877             =f1CLOC EQU    00-45          Clock mode (1 sec update)(-45)
878             =f1EXAC EQU    00-46          EXACT flag                (-46)
879             =f1CMDS EQU    00-47          Command Stack Active      (-47)
880             =f1CTRL EQU    00-48          Control key hit           (-48)
881             *
882             *> Nib Boundary
883             *
884             =f1PWDN EQU    00-49          DSLEEP called from PWRdwn(-49)
885             =f1MKOF EQU    00-50          Req set TRNOF in MAINLP   (-50)
886             =f1TNOF EQU    00-51          Turnoff at MAINLP         (-51)
887             =f1VIEW EQU    00-52          VIEW key pressed          (-52)
888             *
889             *> Nib Boundary
890             *
891             *              00-53          Reserved for Future Use   (-53)
892             *              00-54          Reserved for Future Use   (-54)
893             *              00-55          Reserved for Future Use   (-55)
894             *              00-56          Reserved for Future Use   (-56)
895             *
896             *> Nib Boundary
897             *
898             =f1AC   EQU    00-57          AC Annunciator            (-57)
899             =f1USRX EQU    00-58          User Mode suspend         (-58)
900             =f1RPTD EQU    00-59          Key repeated              (-59)
901             =f1ALRM EQU    00-60          Alarm Annunciator         (-60)
902             *
903             *> Nib Boundary
904             *
905             =f1BAT  EQU    00-61          Low Battery Annunciator   (-61)
```

```
906             =f1PRGM EQU   00-62         Program Annunciator     (-62)
907             =f1SUSP EQU   00-63         Suspend Annunciator     (-63)
908             =f1CALC EQU   00-64         Calc Mode Annunciator   (-64)
909
```

```
                STITLE Token Symbol Definitions
            **============================================================
            **
            **        Token Symbol Definitions for Built-in XROM
            **
            **============================================================

            =XROM01 EQU    #01             Built-in XROM ID

            =tANGLE EQU    (xANGLE)~(XROM01)~(tXFN)   XFN   for ANGLE
            =tMATH  EQU    (xMATH)~(XROM01)~(tXWORD)  XWORD for MATH
            =tEXTND EQU    (xEXTND)~(XROM01)~(tXWORD) XWORD for EXTEND
            =tINTO  EQU    (xINTO)~(XROM01)~(tXWORD)  XWORD for INTO
            =tPCRD  EQU    (xPCRD)~(XROM01)~(tXWORD)  XWORD FOR PCRD
            =tVARS  EQU    (xVARS)~(XROM01)~(tXWORD)  XWORD for VARS
            =tFLOW  EQU    (xFLOW)~(XROM01)~(tXWORD)  XWORD for FLOW
            =tCLOCK EQU    (xCLOCK)~(XROM01)~(tXWORD) XWORD for CLOCK
            =tROUND EQU    (xROUND)~(XROM01)~(tXWORD) XWORD for ROUND
            =tNEAR  EQU    (xNEAR)~(XROM01)~(tXWORD)  XWORD for NEAR
            =tNEG   EQU    (xNEG)~(XROM01)~(tXWORD)   XWORD for NEG
            =tPOS   EQU    (xPOS)~(XROM01)~(tXFN)     XFN   for POS
            =tZERO  EQU    (xZERO)~(XROM01)~(tXWORD)  XWORD for ZERO

            **
            * HP-IL ROM XWORD tokens
            *
            =tINTR  EQU    #15FF                      ON INTR
            =tENTER EQU    #14FF00+tXWORD             ENTER


00000               END
```

```
=BASICs  Abs     181 #000B5 -   790
=CR      Abs  180224 #2C000 -   795
=DISPt   Abs       0 #00000 -   623
=Except  Abs      12 #0000C -   695
 LASTFN  Abs     180 #000B4 -    10   790
=LEEWAY  Abs     212 #000D4 -   788
=LXBFID  Abs    3068 #00BFC -   773
=NoCont  Abs      14 #0000E -   697
=PRINTt  Abs       1 #00001 -   624
=PgmRun  Abs      13 #0000D -   696
=ROMCID  Abs    3070 #00BFE -   775
=Trace   Abs      15 #0000F -   698
=XROM01  Abs       1 #00001 -   917   919   920   921   922   923   924   925
                                926   927   928   929   930   931
=bALTCH  Abs    3067 #00BFB -   771
=bASSGN  Abs    2052 #00804 -   757
=bCARD   Abs    2055 #00807 -   761
=bCHARS  Abs    3067 #00BFB -   772
=bECOMD  Abs    2057 #00809 -   763
=bFIB    Abs    2051 #00803 -   755
=bFILE   Abs    2053 #00805 -   758
=bIEXKY  Abs    2050 #00802 -   753
=bLEX    Abs    3068 #00BFC -   774
=bPILAI  Abs    2064 #00810 -   768
=bPILSV  Abs    2063 #0080F -   767
=bROMTB  Abs    3070 #00BFE -   776
=bSCRTC  Abs    3584 #00E00 -   780
=bSTART  Abs    2056 #00808 -   762
=bSTAT   Abs    2054 #00806 -   759
=bSTMT   Abs    2049 #00801 -   752
=bSTMXQ  Abs    2065 #00811 -   769
=cC->C   Abs     104 #00068 -    82
=cR->C   Abs     105 #00069 -    83
=cRCL    Abs     103 #00067 -    81
=caACTV  Abs      66 #00042 -    44    45
=caCALS  Abs      71 #00047 -    45    46
=caCNTA  Abs      31 #0001F -    37    38
=caCURE  Abs      21 #00015 -    35    36
=caCURS  Abs       6 #00006 -    32    33
=caDATP  Abs      51 #00033 -    41    42
=caERRA  Abs      41 #00029 -    39    40
=caERRS  Abs      36 #00024 -    38    39
=caFSTK  Abs      56 #00038 -    42    43
=caGSTK  Abs      61 #0003D -    43    44
=caINTR  Abs      46 #0002E -    40    41
=caLENG  Abs      84 #00054 -    49
=caPCAD  Abs      26 #0001A -    36    37
=caPMCN  Abs      76 #0004C -    46    47
=caPMTR  Abs      78 #0004E -    47    48
=caPRME  Abs      16 #00010 -    34    35
=caPRMS  Abs      11 #0000B -    33    34
=caRTNT  Abs      83 #00053 -    48    49
=dCARD   Abs       7 #00007 -    93
=dIRAM   Abs       1 #00001 -    91
=dMAIN   Abs       0 #00000 -    90
```

```
=dPCRD   Abs       7 #00007 -    94
=dPORT   Abs       1 #00001 -    92
=fAOS    Abs     223 #000DF -   402
=fASCII  Abs       1 #00001 -   389
=fBASIC  Abs   57876 #0E214 -   388
=fBIN    Abs   57860 #0E204 -   395
=fDATA   Abs   57584 #0E0F0 -   394
=fEOF    Abs     255 #000FF -   400
=fEOR    Abs     239 #000EF -   401
=fEOS    Abs     111 #0006F -   405
=fKEY    Abs   57868 #0E20C -   390
=fLEX    Abs   57864 #0E208 -   396
=fLIF1   Abs       1 #00001 -   391
=fMOS    Abs     127 #0007F -   404
=fSDATA  Abs   57552 #0E0D0 -   393
=fSOS    Abs     207 #000CF -   403
=fTEXT   Abs       1 #00001 -   392
=f1AC    Abs     -57 #FFFC7 -   898
=f1ALRM  Abs     -60 #FFFC4 -   901
=f1BASE  Abs     -16 #FFFF0 -   836
=f1BAT   Abs     -61 #FFFC3 -   905
=f1BEEP  Abs      -2 #FFFFE -   813
=f1BPLD  Abs     -25 #FFFE7 -   854
=f1CALC  Abs     -64 #FFFC0 -   908
=f1CLOC  Abs     -45 #FFFD3 -   877
=f1CMDS  Abs     -47 #FFFD1 -   879
=f1CTON  Abs      -3 #FFFFD -   814
=f1CTRL  Abs     -48 #FFFD0 -   880
=f1DG0   Abs     -17 #FFFEF -   840
=f1DG1   Abs     -18 #FFFEE -   841
=f1DG2   Abs     -19 #FFFED -   842
=f1DG3   Abs     -20 #FFFEC -   843
=f1DORM  Abs     -43 #FFFD5 -   872
=f1DVZ   Abs      -7 #FFFF9 -   821
=f1EOT   Abs     -23 #FFFE9 -   849
=f1EXAC  Abs     -46 #FFFD2 -   878
=f1EXTD  Abs     -22 #FFFEA -   848
=f1FXEN  Abs     -13 #FFFF3 -   833
=f1INFR  Abs     -11 #FFFF5 -   828
=f1INX   Abs      -4 #FFFFC -   815
=f1IVL   Abs      -8 #FFFF8 -   822
=f1LC    Abs     -15 #FFFF1 -   835
=f1MKOF  Abs     -50 #FFFCE -   885
=f1MPI   Abs     -41 #FFFD7 -   865
=f1NEGR  Abs     -12 #FFFF4 -   829
=f1NOFN  Abs     -42 #FFFD6 -   871
=f1NOPR  Abs     -26 #FFFE6 -   855
=f1NZ4   Abs     -24 #FFFE8 -   850
=f1OVF   Abs      -6 #FFFFA -   820
=f1PDWN  Abs     -21 #FFFEB -   847
=f1PRGM  Abs     -62 #FFFC2 -   906
=f1PWDN  Abs     -49 #FFFCF -   884
=f1QIET  Abs      -1 #FFFFF -   812
=f1RAD   Abs     -10 #FFFF6 -   827
=f1RPTD  Abs     -59 #FFFC5 -   900
```

=flRTN	Abs	-44	#FFFD4 -	873				
=flSCEN	Abs	-14	#FFFF2 -	834				
=flSUSP	Abs	-63	#FFFC1 -	907				
=flTNOF	Abs	-51	#FFFCD -	886				
=flUNF	Abs	-5	#FFFFB -	819				
=flUSER	Abs	-9	#FFFF7 -	826				
=flUSRX	Abs	-58	#FFFC6 -	899				
=flVIEW	Abs	-52	#FFFCC -	887				
=lACCSb	Abs	1	#00001 -	352	371			
=lAp	Abs	16	#00010 -	671				
=lBPOSp	Abs	5	#00005 -	663				
=lCOPYb	Abs	1	#00001 -	351	370			
=lCPOSb	Abs	6	#00006 -	359	379			
=lD0p	Abs	5	#00005 -	668				
=lD1p	Abs	5	#00005 -	669				
=lDATEh	Abs	6	#00006 -	196				
=lDBEGb	Abs	11	#0000B -	356	376			
=lDEVC	Abs	5	#00005 -	647				
=lDEVCb	Abs	1	#00001 -	353	372			
=lDLENb	Abs	6	#00006 -	360	380			
=lDp	Abs	16	#00010 -	670				
=lEOL	Abs	2	#00002 -	206				
=lFBEGb	Abs	6	#00006 -	354	374			
=lFBF#b	Abs	3	#00003 -	348	367			
=lFIB	Abs	63	#0003F -	363				
=lFIL#b	Abs	2	#00002 -	347	366			
=lFILSV	Abs	50	#00032 -	643				
=lFLAGh	Abs	2	#00002 -	194				
=lFLENh	Abs	5	#00005 -	197				
=lFNAM+	Abs	4	#00004 -	646				
=lFNAM8	Abs	16	#00010 -	645				
=lFNAMh	Abs	16	#00010 -	192				
=lFSIZb	Abs	6	#00006 -	362				
=lFTYPb	Abs	4	#00004 -	349	368			
=lFTYPh	Abs	4	#00004 -	193				
=lLXADR	Abs	5	#00005 -	480	481			
=lLXENT	Abs	11	#0000B -	481				
=lLXFAD	Abs	5	#00005 -	434				
=lLXID	Abs	2	#00002 -	432	481			
=lLXTKR	Abs	4	#00004 -	433	481			
=lMSGp	Abs	4	#00004 -	438	449	460	474	
=lPOL#p	Abs	5	#00005 -	665				
=lPOLLp	Abs	5	#00005 -	439	449	460	474	
=lPOLSV	Abs	62	#0003E -	683				
=lPOLra	Abs	6	#00006 -	684				
=lPROTb	Abs	1	#00001 -	350	369			
=lREC#b	Abs	4	#00004 -	357	377			
=lRECLb	Abs	4	#00004 -	358	378			
=lRLENb	Abs	5	#00005 -	361	381			
=lRTN1p	Abs	5	#00005 -	664				
=lRTN2p	Abs	5	#00005 -	666				
=lRTN3p	Abs	5	#00005 -	667				
=lSHLNb	Abs	2	#00002 -	355	375			
=lSPDTB	Abs	78	#0004E -	440	449			
=lSPDn	Abs	1	#00001 -	435	449			

```
=lSPDn2  Abs       1 #00001 -   436   449   460
=lTEXTp  Abs       4 #00004 -   437   449   460
=lTIMEh  Abs       4 #00004 -   195
=o41sod  Abs       5 #00005 -   205
=oACCSb  Abs      11 #0000B -   370   371
=oAp     Abs      62 #0003E -   681   683
=oBNsod  Abs      17 #00011 -   200
=oBPOSp  Abs       5 #00005 -   673
=oBSsod  Abs      17 #00011 -   201
=oCOPYb  Abs      10 #0000A -   369   370
=oCPOSb  Abs      40 #00028 -   378   379
=oDOp    Abs      25 #00019 -   678
=oD1p    Abs      30 #0001E -   679
=oDATEh  Abs      26 #0001A -   184
=oDAsod  Abs      13 #0000D -   202
=oDBEGb  Abs      21 #00015 -   375   376
=oDEVCb  Abs      12 #0000C -   371   372
=oDLENb  Abs      46 #0002E -   379   380
=oDp     Abs      46 #0002E -   680
=oFBEGb  Abs      13 #0000D -   372   374
=oFBF#b  Abs       2 #00002 -   366   367
=oFIL#b  Abs       0 #00000 -   365   366
=oFLAGh  Abs      20 #00014 -   182
=oFLENh  Abs      32 #00020 -   185   190
=oFLSTr  Abs      49 #00031 -   188
=oFNAMh  Abs       0 #00000 -   180
=oFSIZb  Abs      57 #00039 -   381
=oFT-FL  Abs      16 #00010 -   190
=oFTYPb  Abs       5 #00005 -   367   368
=oFTYPh  Abs      16 #00010 -   181   190
=oIMPLh  Abs      37 #00025 -   186
=oKYsod  Abs       5 #00005 -   199
=oLXsod  Abs       5 #00005 -   203
=oMAINT  Abs      93 #0005D -   449
=oMSGPT  Abs       9 #00009 -   474
=oPOL#p  Abs      10 #0000A -   675
=oPROTb  Abs       9 #00009 -   368   369
=oREC#b  Abs      32 #00020 -   376   377
=oRECLb  Abs      36 #00024 -   377   378
=oRLENb  Abs      52 #00034 -   380   381
=oRTN1p  Abs      10 #0000A -   674
=oRTN2p  Abs      15 #0000F -   676
=oRTN3p  Abs      20 #00014 -   677
=oSHLNb  Abs      19 #00013 -   374   375
=oSPDTB  Abs     273 #00111 -   467
=oSPDn2  Abs      14 #0000E -   460
=oSUBLn  Abs      37 #00025 -   187
=oTIMEh  Abs      22 #00016 -   183
=oTXsod  Abs       5 #00005 -   204
=pBSCen  Abs     245 #000F5 -   605
=pBSCex  Abs     246 #000F6 -   606
=pCALRS  Abs      54 #00036 -   584
=pCALSV  Abs      55 #00037 -   585
=pCAT    Abs       6 #00006 -   507
=pCAT$   Abs       7 #00007 -   508
```

```
=pCLDST  Abs     255 #000FF -   615
=pCMPLX  Abs      56 #00038 -   589
=pCONFG  Abs     251 #000FB -   611
=pCOPYx  Abs       8 #00008 -   509
=pCRDAB  Abs      51 #00033 -   578
=pCREAT  Abs       9 #00009 -   510
=pCRT=8  Abs      35 #00023 -   549
=pCURSR  Abs      41 #00029 -   561
=pDATLN  Abs      42 #0002A -   562
=pDEVCp  Abs       1 #00001 -   498
=pDIDST  Abs      10 #0000A -   511
=pDSWKY  Abs     253 #000FD -   613
=pDSWNK  Abs     254 #000FE -   614
=pEDIT   Abs      43 #0002B -   564
=pENTER  Abs      18 #00012 -   519
=pEOFIL  Abs      37 #00025 -   554
=pERROR  Abs     242 #000F2 -   602
=pExcpt  Abs     248 #000F8 -   608
=pFASCH  Abs      44 #0002C -   565
=pFILDC  Abs       2 #00002 -   499
=pFILXQ  Abs       3 #00003 -   500
=pFINDF  Abs      23 #00017 -   528
=pFNIN   Abs      61 #0003D -   594
=pFNOUT  Abs      62 #0003E -   595
=pFPROT  Abs      11 #0000B -   512
=pFSPCp  Abs       4 #00004 -   502
=pFSPCx  Abs       5 #00005 -   503
=pFTYPE  Abs      45 #0002D -   566
=pIMCHR  Abs      30 #0001E -   541
=pIMXCH  Abs      31 #0001F -   542
=pIMXQT  Abs      29 #0001D -   540
=pIMbck  Abs      32 #00020 -   543
=pIMcpi  Abs      33 #00021 -   544
=pIMcpw  Abs      34 #00022 -   545
=pKYDF   Abs      27 #0001B -   535
=pLIST   Abs      12 #0000C -   513
=pLIST2  Abs      46 #0002E -   567
=pMEM    Abs     241 #000F1 -   601
=pMERGE  Abs      13 #0000D -   514
=pMNLP   Abs     250 #000FA -   610
=pMRGE2  Abs      47 #0002F -   568
=pPARSE  Abs     244 #000F4 -   604
=pPRGPR  Abs      50 #00032 -   574
=pPRIN#  Abs      38 #00026 -   555
=pPRTCL  Abs      14 #0000E -   515
=pPRTIS  Abs      15 #0000F -   516
=pPURGE  Abs      16 #00010 -   517
=pPWROF  Abs     252 #000FC -   612
=pRCRD   Abs      52 #00034 -   579
=pRDCBF  Abs      24 #00018 -   529
=pRDNBF  Abs      25 #00019 -   530
=pREAD#  Abs      39 #00027 -   556
=pREN    Abs      57 #00039 -   590
=pRNAME  Abs      17 #00011 -   518
=pRTNTp  Abs      58 #0003A -   591
```

```
=pRUNft  Abs      48 #00030 -   569
=pRUNnB  Abs      49 #00031 -   570
=pSREC#  Abs      40 #00028 -   557
=pSREQ   Abs     249 #000F9 -   609
=pTEST   Abs     240 #000F0 -   600
=pTIMR#  Abs      59 #0003B -   592
=pTRANS  Abs     239 #000EF -   599
=pTRFMx  Abs      60 #0003C -   593
=pVER$   Abs       0 #00000 -   494
=pWARN   Abs     243 #000F3 -   603
=pWCRD   Abs      53 #00035 -   580
=pWCRD8  Abs      36 #00024 -   550
=pWRCBF  Abs      26 #0001A -   531
=pWTKY   Abs      28 #0001C -   536
=pZERPG  Abs     247 #000F7 -   607
=sARITH  Abs       7 #00007 -   728
=sBYEx   Abs       0 #00000 -   703
=sCARD   Abs       2 #00002 -   710
=sCARDC  Abs       8 #00008 -   731
=sCHAIN  Abs      11 #0000B -   739
=sCONT   Abs      10 #0000A -   737
=sCONTK  Abs       9 #00009 -   735
=sCURBT  Abs       3 #00003 -   715
=sCURUD  Abs       4 #00004 -   718
=sCURUP  Abs       2 #00002 -   711
=sDEST   Abs       3 #00003 -   716
=sENDx   Abs       1 #00001 -   707
=sEOF    Abs       7 #00007 -   729
=sERROR  Abs       0 #00000 -   704
=sEXTDV  Abs       0 #00000 -   705
=sEXTGS  Abs       5 #00005 -   722
=sGOSUB  Abs       3 #00003 -   717
=sIRAM   Abs       2 #00002 -   712
=sKEYS   Abs       5 #00005 -   723
=sMAINc  Abs       5 #00005 -   724
=sNoChn  Abs       2 #00002 -   713
=sONERR  Abs       4 #00004 -   719
=sONTMR  Abs       6 #00006 -   725
=sPCRD   Abs       8 #00008 -   732
=sREADI  Abs       4 #00004 -   720
=sRENAM  Abs       6 #00006 -   726
=sRENUM  Abs       8 #00008 -   733
=sRESTR  Abs      10 #0000A -   738
=sRETRN  Abs       0 #00000 -   706
=sRFILE  Abs       8 #00008 -   734
=sRUNBn  Abs       4 #00004 -   721
=sRUNDC  Abs       7 #00007 -   730
=sSST    Abs       2 #00002 -   714
=sSSTdc  Abs       1 #00001 -   708
=sSTAT   Abs       6 #00006 -   727
=sUNDEF  Abs       1 #00001 -   709
=sXWORD  Abs       9 #00009 -   736
=tANGLE  Abs  393651 #601B3 -   919
=tCLOCK  Abs 1376751 #501EF -   926
=tENTER  Abs 1376239 #4FFEF -   937
```

```
=tEXTND  Abs 2490863 #601EF -   921
=tFLOW   Abs 2687471 #901EF -   925
=tINTO   Abs 3015151 #E01EF -   922
=tINTR   Abs    5631 #015FF -   936
=tMATH   Abs 3539439 #601EF -   920
=tNEAR   Abs 3932655 #C01EF -   928
=tNEG    Abs 3998191 #D01EF -   929
=tPCRD   Abs 4063727 #E01EF -   923
=tPOS    Abs 4325811 #201B3 -   930
=tROUND  Abs 4981231 #C01EF -   927
=tVARS   Abs 5964271 #B01EF -   924
 tXFN    Abs     179 #000B3 -    10   919   930
 tXWORD  Abs     239 #000EF -    10   920   921   922   923   924   925   926
                                 927   928   929   931   937
=tZERO   Abs 1835503 #C01EF -   931
=ufCHN#  Abs     103 #00067 -    69    70
=ufCNTA  Abs      16 #00010 -    57    58
=ufFSTK  Abs      83 #00053 -    64    65
=ufGSTK  Abs      88 #00058 -    65    66
=ufLENG  Abs     106 #0006A -    71
=ufMSTK  Abs      78 #0004E -    63    64
=ufPC    Abs      11 #0000B -    56    57
=ufPMCN  Abs      93 #0005D -    66    67
=ufPMTR  Abs      95 #0005F -    67    68
=ufRSTK  Abs      26 #0001A -    59    60
=ufRTNA  Abs       6 #00006 -    55    56
=ufRTNT  Abs     105 #00069 -    70    71
=ufSR-0  Abs      46 #0002E -    61    62
=ufSR-1  Abs      62 #0003E -    62    63
=ufSTM0  Abs      21 #00015 -    58    59
=ufSTM1  Abs      41 #00029 -    60    61
=ufSTSV  Abs     100 #00064 -    68    69
 xANGLE  Abs       6 #00006 -    11   919
 xCLOCK  Abs      21 #00015 -    11   926
 xEXTND  Abs      38 #00026 -    11   921
 xFLOW   Abs      41 #00029 -    11   925
 xINTO   Abs      46 #0002E -    11   922
 xMATH   Abs      54 #00036 -    11   920
 xNEAR   Abs      60 #0003C -    11   928
 xNEG    Abs      61 #0003D -    11   929
 xPCRD   Abs      62 #0003E -    11   923
 xPOS    Abs      66 #00042 -    11   930
 xROUND  Abs      76 #0004C -    11   927
 xVARS   Abs      91 #0005B -    11   924
 xZERO   Abs      28 #0001C -    11   931
```

Input Parameters

  Source file name is TI&EQU::MS

  Listing file name is TI/EQU:TI:ML::-1

  Object file name is TI%EQU:TI:MS::-1

```
                                  111111
                        0123456789012345
```

  Initial flag settings are

Errors

  None

Saturn Assembler News

Scan Copyright ©
The Museum of HP Calculators
www.hpmuseum.org

Original content used with permission.

Thank you for supporting the Museum of HP Calculators by purchasing this Scan!

Please to not make copies of this scan or make it available on file sharing services.